

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Alternative Solutions for Substation Telecontrol

Hugo Filipe Oliveira Rocha

FINAL VERSION

Masters in Electrical and Computers Engineering

Adviser: Maria Teresa Andrade Assistant Professor

Co-adviser: Pedro José Fonseca Engineer

July 26, 2013

Resumo

Os serviços de distribuição eléctrica cobrem grandes áreas geográficas e expandem-se ao longo de grandes distâncias. Os operadores da rede necessitam de métodos fiáveis e financeiramente suportáveis para aceder às subestações de modo a garantir a monitorização remota. O acesso remoto é normalmente garantido recorrendo a fibra ótica ou a feixes hertzianos, no entanto estas tecnologias têm as suas limitações. Existe a necessidade de encontrar novas soluções alternativas que garantam o controlo remoto aplicado a situações reais. Esta dissertação procura explorar novas maneiras de garantir a comunicação de dados em sistemas SCADA. Estas soluções distanciam-se das tecnologias clássicas explorando alternativas que recorrem a modems wireless para garantir o telecontrolo de subestações aplicada na maior empresa de distribuição de energia em Portugal.

Esta dissertação detalha todo o trabalho desenvolvido para implementar a solução, a sua integração e testes e outras funcionalidades extra que foram desenvolvidas. A solução revela-se como uma solução flexível e adequada para o telecontrolo de subestações.

Abstract

Electric power distribution services often cover large areas and reach great distances. The distribution network includes, among other equipment, remotely located substations, in charge mainly of performing transformations from high to low voltage and vice-versa. Normally, the substations are unattended but given the crucial role they play within the distribution of electricity to the final consumer, it is essential that reliable and cost-effective monitoring systems are made available to remotely located operators. The remote access is traditionally provided by SCADA systems (Supervisory Control And Data Acquisition), supported by optical fiber or microwave links. However, these communication technologies have some limitations and therefore there is the need to research alternative solutions that may be able to guarantee efficient remote monitoring in practical conditions. This dissertation aims to explore new alternative ways to implement substation telecontrol in SCADA systems within the distribution network of one of the leading electric power distribution company in Portugal. These solutions detach from the classic ways to transmit telemetry data by exploring the use of wireless modems applied to the telecontrol of substations. This manuscript details all the work developed during the course of this dissertation to implement the sought solution, including its integration with existing equipment, as well as the execution of functional and performance tests. It also describes additional functionality that was designed and implemented, providing the final system with flexibility and capabilities that were not foreseen at the beginning. According to the feedback obtained from the tests performed, the solution has revealed itself as an efficient, cost-effective and flexible solution for substation telecontrol.

Acknowledgements

Although this is an individual work I would like to show my appreciation and gratitude to all those who supported me during this work.

Therefore I would like to thank to my supervisor Maria Teresa Andrade for all the helpful advices, guidance and for the time spent reviewing the dissertation's content.

I would like to thank to my co-adviser Pedro José Fonseca for his help and guidance, for facilitating all the required hardware and many important information that lead to the successful accomplishment of the proposed objectives.

I would like to thank to the EDP's ATOM department particularly to the technicians Fernando Rocha and Rui Ferreira for their help in understanding crucial hardware details and to the engineer Pedro Vidal for his restless support.

Finally I would like to thank to my friends especially Hugo Alves and Rita Neves for their support and everlasting friendship. To my girlfriend Eduarda Sousa for finding me when all seemed lost, being more than I could ever wish for and being my answer to the hardest question in life. And to my mother, brother and especially to my father Nelson Rocha, who besides my father, is a close friend, who always did for me more than one could ever expect, for his unwavering dedication and everlasting support, for always doing the impossible to help me with this project and life itself, for being someone who will always have my gratitude and who I will always be grateful for.

Hugo Filipe Oliveira Rocha

*“Quality is never an accident; it is always the result of high intention,
sincere effort, intelligent direction and skillful execution;
it represents the wise choice of many alternatives.”*

William A. Foster

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Scope	2
1.3	Objectives and Outcomes	2
1.4	Structure	3
2	State of the Art	5
2.1	SCADA Systems	5
2.2	SCADA Protocols	6
2.2.1	OSI Model	6
2.2.2	Modbus Protocol	8
2.2.3	DNP 3 Protocol	9
2.2.4	IEC 60870-5 Standard	10
2.2.5	Protocol Comparison	14
2.3	GPRS Technology	16
2.3.1	Siemens TC65	17
2.3.2	Cinterion MC55i	17
2.3.3	iTegno 39XX	17
2.3.4	Robustel GoRugged M1000	17
2.3.5	ABB RER601 and RER603	17
2.4	SCADA Satellite Systems Providers	18
2.4.1	Orbcomm	18
2.4.2	Thuraya	18
2.4.3	Iridium Communications	18
2.4.4	Inmarsat	19
2.4.5	Globalstar	19
2.5	Overview	19
3	Specification of the adopted solution	21
3.1	Objectives and Functionalities	21
3.2	System's Architecture	22
3.3	System's Specification	23
3.4	Impact in the Company	31
4	Development	33
4.1	System Overview	33
4.2	Core Functionalities	34
4.2.1	Initiation	34

4.2.2	Reception	35
4.2.3	Transmission	37
4.3	Additional Features	37
4.3.1	Error Correction Mechanisms	37
4.3.2	Operation Modes	42
4.3.3	Dynamical Variable Assignment Mechanisms	42
4.4	Online Monitoring Interface	44
4.4.1	REST Interface Design	45
4.4.2	Interface Functionalities	46
4.5	Performance Considerations	53
4.5.1	J2ME and Jar Size	54
4.5.2	Busy Loops	56
4.5.3	Garbage Collection	57
4.5.4	Multi-threading	59
4.5.5	IEC 60870 Frame Validation	59
4.5.6	TCP vs. UDP	60
4.5.7	HTTP Requests Minimization	64
4.5.8	Web Integration Protocol Efficiency	64
4.5.9	Automatic Modem's Geographic Location Detection	67
4.6	Satellite Systems Benchmarking	69
4.6.1	Iridium	70
4.6.2	Inmarsat	70
4.6.3	Orbcomm	71
4.6.4	Thuraya	71
4.6.5	Globalstar	71
4.6.6	Overview	72
5	Conclusions and Tests	75
5.1	Garbage Collection	75
5.2	UDP and TCP	78
5.3	Real Tests	80
5.4	Conclusions	80
5.5	Limitations	82
5.6	Future Work	82
A	Garbage Collection Tests	85
A.1	String Object Tests	85
A.2	StringBuffer Object Tests	88
B	TCP and UDP Tests	93
B.1	UDP Tests	93
B.2	TCP Tests	95
C	Real Tests Logs	99
C.1	Log Sample	99
D	Use Case Textual Specification	103
D.1	Use Case Textual Specification	103

CONTENTS

xi

References

109

List of Figures

2.1	Typical SCADA hierarchy	6
2.2	Typical SCADA system	7
2.3	OSI Model's Structure	8
2.4	IEC 60870 Structure	11
2.5	IEC 60870 frame types	12
2.6	IEC 60870 link user data detail	13
2.7	Message sequence diagram	14
3.1	Overall system's architecture	23
3.2	System's Collaboration Diagram	24
3.3	Technician's use case diagram	25
3.4	Application's main use case diagram	26
3.5	Java application class diagram	30
4.1	Java application's block diagram	34
4.2	High level applications' initiation flow chart	36
4.3	Reception thread's high level flowchart	38
4.4	Transmission thread's high level flow chart	39
4.5	Configuration interface	43
4.6	Client application's file structure	44
4.7	Application's web request time sequence	45
4.8	Online modems listing	46
4.9	Modem's geographic location	47
4.10	Modem's details	47
4.11	Modem's sent and received frames graph	48
4.12	Online SIM's listing	48
4.13	Online configuration's listing	49
4.14	Modem's configuration interface	49
4.15	Active connections listing	50
4.16	Active connections details	50
4.17	Active connections geographic location	51
4.18	Active connections reset states	51
4.19	Predefined connections listing	52
4.20	Insert predefined connection interface	52
4.21	Events visualization	53
4.22	Event filtering	54
4.23	Solution's occupied space comparison	55
4.24	Occupied space after size reduction techniques	56

4.25	Source code before performance tuning	58
4.26	Source code after performance tuning	58
4.27	Thread wait method implementation	59
4.28	Initial frame validation sequence	61
4.29	Frame validation's state machine	62
4.30	Initiation HTTP requests before optimization	65
4.31	Initiation HTTP requests after optimization	66
4.32	Example of an application's XML object	66
4.33	Example of the implemented custom protocol	67
4.34	Mobile network geographic structure	68
5.1	TCP/UDP tests scheme	78
5.2	Frontend's real test configuration	81

List of Tables

2.1	IEC 60870, DNP3 and Modbus comparison	16
3.1	Exchange data use case textual description	25
3.2	Exchange data course of events	26
3.3	Create connection use case textual description	27
3.4	Create connection course of events	27
3.5	Register use case textual description	27
3.6	Register course of events	28
3.7	Register connection use case textual description	28
3.8	Register connection course of events	28
3.9	Report information use case textual description	29
3.10	Report information course of events	29
4.1	J2ME TCP/UDP comparison	63
4.2	Geographic location detection method comparison	69
4.3	Main satellite services airtime charges	72
4.4	Main satellite services monthly airtime cost for a given amount of data	72
5.1	Garbage collection tests sample size estimation parameters	76
5.2	Garbage collection tests sample size estimation results	76
5.3	Garbage collection tests results	77
5.4	Garbage collection paired t-test results	77
5.5	TCP/UDP tests sample size estimation parameters	78
5.6	TCP/UDP tests sample size estimation results	79
5.7	TCP/UDP tests results	79
5.8	TCP/UDP paired t-test results	79
A.1	String object measured delay	85
A.2	StringBuffer object measured delay	88
B.1	UDP connection measured delay	93
B.2	TCP connection measured delay	95
D.1	Configure serial port parameters use case textual description	103
D.2	Configure server use case textual description	104
D.3	Configure operation mode use case textual description	104
D.4	Configure application's parameters use case textual description	105
D.5	Generate configuration file use case textual description	105
D.6	Configure connections use case textual description	106

D.7 View modems' list and details use case textual description	106
D.8 View SIM cards' list and details use case textual description	107
D.9 View connections use case textual description	107
D.10 Reset modem use case textual description	108

Symbols and Abbreviations

ASDU	Application Service Data Unit
ATOM	Automation, Telecontrol, Operations and Maintenance
DNP	Distributed Network Protocol
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ISDN	Integrated Services Digital Network
ISO	International Organization of Standardization
J2ME	Java 2 Platform, Micro Edition
JVM	Java Virtual Machine
M2M	Machine to Machine
MCC	Mobile Country Code
MNC	Mobile Network Code
OSI	Open System Interconnection
PDP	Packet Data Protocol
PLC	Programmable Logic Controller
REST	Representational State Transfer
RTU	Remote Terminal Unit
RUDICS	Router Unrestricted Digital Internetworking Connectivity Solution
SCADA	Supervisory Control and Data Acquisition
SCP	Secure Copy Protocol
SOAP	Simple Object Access Protocol
SSH	Secure File Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Chapter 1

Introduction

This introductory chapter provides a bird's eye view of the work that was developed within the context of this dissertation. It presents the main objectives that were established, a concise overview of the context and scope of the work, identifying the main obstacles to overcome as well as the motivation to surpass those difficulties. It briefly indicates the main outcomes of the developed work, highlighting strengths and weaknesses. Finally it also describes how this document is organized, listing the different chapters that compose it and the main topics addressed in each one.

1.1 Motivation

In modern industrial processes and electrical distribution systems telemetry is often needed to connect equipment and systems separated by large distances. This can range from a few meters to thousands of kilometers. Telemetry is usually used to send commands and receive monitoring information from these remote locations. SCADA often includes telemetry and data acquisition and involves information gathering, transferring it back to the central station, carrying out any necessary analysis and control and then displaying that information on a number of operator screens [1]. The operational efficiency of any commercial service based on SCADA, greatly depends on the ability to remotely control and monitor devices and equipment located in distant places, such as machines or complete substations. It is critical for the operations and maintenance departments, to have tools that can be easily deployed, enabling to quickly access the status of remote equipment and extract useful information. The ability to obtain this data through remote access is a prerequisite [2]. In many services, like electric power or water distribution, it is necessary to install equipment in distant places and geographically sparse. Nowadays it is unacceptable to maintain individuals in every substation, mostly due to the high costs involved, highlighting the importance of remote monitoring through telecontrol. An efficient and competitive service must necessarily rely on the use of remote diagnosis tools to detect in real-time faulty equipment and obtain relevant information to accordingly initiate maintenance tasks. These features greatly reduce the operation costs and time efficiency of the technicians. The research of viable alternatives to achieve these objectives applied to the context of crucial services has a great importance to their success [3].

SCADA uses a centralized system to monitor and control processes spread out over large geographic locations. Telemetry is used to receive monitoring information or to send commands to these locations. It is commonly found in industrial processes, namely, electrical power generation and transmission, water treatment and distribution, oil and gas pipelines, wind farms and others [1, 2]. SCADA systems traditionally use radio communications (VHF/UHF or microwave radio), direct wired communications (copper, coaxial or fiber optic cable) or a combination of the two. However these methods are very vulnerable when major natural disasters occur or even too costly or difficult to implement in some locations. More than often, a bulldozer rips a fiber optic cable, bad weather makes radio links fail or other random accidents lead to loss of communication. It is essential to research alternative methods that can overcome the limitations of the usual communication technologies and guarantee the availability of communication links while repairs are being done. GPRS technology and satellite systems applied to SCADA may offer a powerful solution to implement the communications between geographically sparse systems, constituting suitable alternatives to the usual communication methods. Investigating new telecontrol alternatives and how to efficiently deploy them, can lead to great cost reductions and time optimization, operational efficiency and overall system performance of any service. These objectives highlight the importance of the research of new solutions and new alternatives to the telecontrol and communication of SCADA information.

1.2 Scope

This dissertation was proposed by EDP, an electrical distribution and production company, aiming at exploring new alternatives to implement the telecontrol of their substations. Specifically, the goals established for this dissertation are to evaluate the possibility of using GPRS technology and satellite communications to implement the remote control within the SCADA system, consequently developing a prototype and integrating it within the EDP's system. The corresponding work will be developed at specific department within EDP (ATOM), responsible for handling the operations and maintenance of their communications and equipment. It will be developed in Gaia at the EDP's installations, under the guidance of the engineer Pedro José Fonseca. Internally the dissertation will be supervised by professor Maria Teresa Andrade.

1.3 Objectives and Outcomes

The overall goal of this dissertation is to investigate and recommend flexible and efficient alternatives for the implementation of the remote control systems of EDP's substations. The work associated to this dissertation has been conducted at the EDP's Operations and Maintenance department building, which has proved to be beneficial for the successful achievement of the proposed work. This has in particular facilitated the fulfillment of the first set of objectives that had been established, namely, to get acquainted with the activities of the department, to study the GPRS technology, the IEC 60870-5 standard, as well as their applications. The second set

of objectives included investigating the potential of using GPRS technology to transmit SCADA information, consequently determining the best alternatives to implement such a solution for the telecontrol of EDP's substations. Based on the outcomes of this phase, the next objective was to actually implement a solution adopting the identified optimal approach. This has involved the specification and development of a Java-based application to be deployed on GPRS modems. Such application had to support the initially studied protocols to transfer SCADA information between SCADA frontends, located at the main operational center, and the remote substations. The final set of objectives established for this dissertation was to investigate the feasibility and eventual benefits, of using satellite communications as an alternative to GPRS.

The outcome of the work conducted based on this set of proposed objectives was an automated software system that supports the exchange and transmission of IEC 60870 data between a substation and frontend. The system also features several error recovery techniques and performance tuning. The online interface is capable to remotely and automatically register, manage, monitor and configure every modem and the modem network.

1.4 Structure

This manuscript is divided into five main parts. The present chapter has provided a bird's eye view of the dissertation. Chapter 2 provides a description of the review performed on the relevant state of the art, notably, on the most important protocols to be used, on the available technologies to allow developing the functionality necessary to fulfill the objectives, on the services available to transmit telemetry data through satellite communications and the main GPRS modems available. The functional specification of the adopted solution, together with its overall architecture, is described in chapter 3. It identifies the functionality offered by the system to support the initially proposed objectives, presenting a formal description of the system through the use of various formal diagrams, namely, use cases, component, collaboration and class diagrams. Chapter 4 describes in detail all the work that was undertaken to develop the identified solution. It provides a description of the main functionality implemented, also describing operational aspects of the system, indicating how the system operates to deliver the referred functionality. This chapter also addresses performances issues of the system, indicating approaches tested and alternatives implemented to overcome initial limitations. Chapter 5 describes operational and usability tests of the system in a real-world environment, presenting the obtained results and draws some conclusions and briefly presents future work that could be performed based on the outcomes of this dissertation and on additionally identified needs of the user of the developed technology.

Chapter 2

State of the Art

This chapter provides relevant background concerning technologies that are foreseen to be used in the development of the project and their main alternatives. It will be presented an overview of the main protocols, hardware and solutions relevant to the project. It will be described the main GPRS modems available and their features, an overview of the IEC 60870-5 standard and its main alternatives and the main satellite solutions for telemetry in Europe.

2.1 SCADA Systems

In modern industrial processes it's often crucial to connect and transmit data from equipments and systems separated by great distances [4]. These distances can range from few meters to thousands of kilometers. Telemetry is used to send programs, data or commands and receive monitoring and measurement data from these equipments. SCADA typically uses telemetry and data acquisition to collect information from the equipments, transmitting it to the central site, manage the processing of data and display the information in several screens or any other user interfaces [5]. It also manages the sending of commands and control actions back to the equipments. The figure 2.1 describes a typical SCADA hierarchy.

The systems usually consist of several equipments or RTUs collecting field data and transmitting it back to the central station through a communications system. The communications system can be implemented with wire, fiber optic, radio links or even satellite. To optimize the communication of data, different protocols can be used to guarantee the error detection and efficiency of the link. The central station displays the data and allows the user to send commands and control the devices remotely [1]. This leads to safer operations, a more efficient and reliable system, lower costs of operation and to the optimization of the industry's operation and processes. Figure 2.2 pictures a typical SCADA system with different communication technologies.

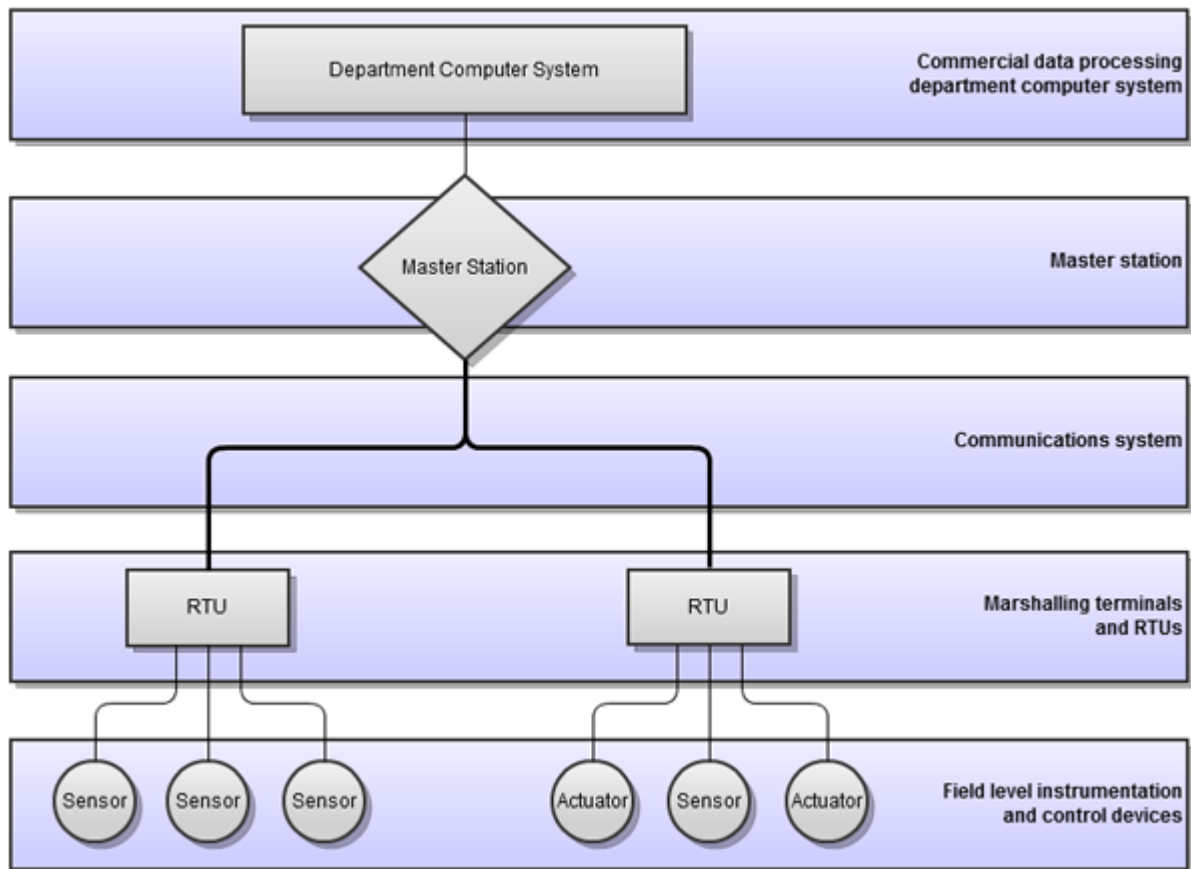


Figure 2.1: Typical SCADA hierarchy

2.2 SCADA Protocols

A protocol is a set of formal rules to describe how information should be transmitted inside a network. These rules can be low level, defining physical and electrical characteristics and bit level operations, or high level, describing for example the frame organization and content [6]. In this chapter it will be provided a brief overview of the OSI model and the main functions and fundamentals of the main SCADA protocols used in electrical systems: Modbus, DNP3 and IEC 60870.

2.2.1 OSI Model

All modern protocols have a representation in the OSI model. This model is an ISO standard with seven layers. All the layers define the structure of a communications network to be developed by each protocol. The flow of information is done vertically; each layer receives the information of the layer directly above and passes the information to the layer directly below. The information of each layer is destined to the same layer on the receiving system [7]. The structure of the model is illustrated in image 2.3.

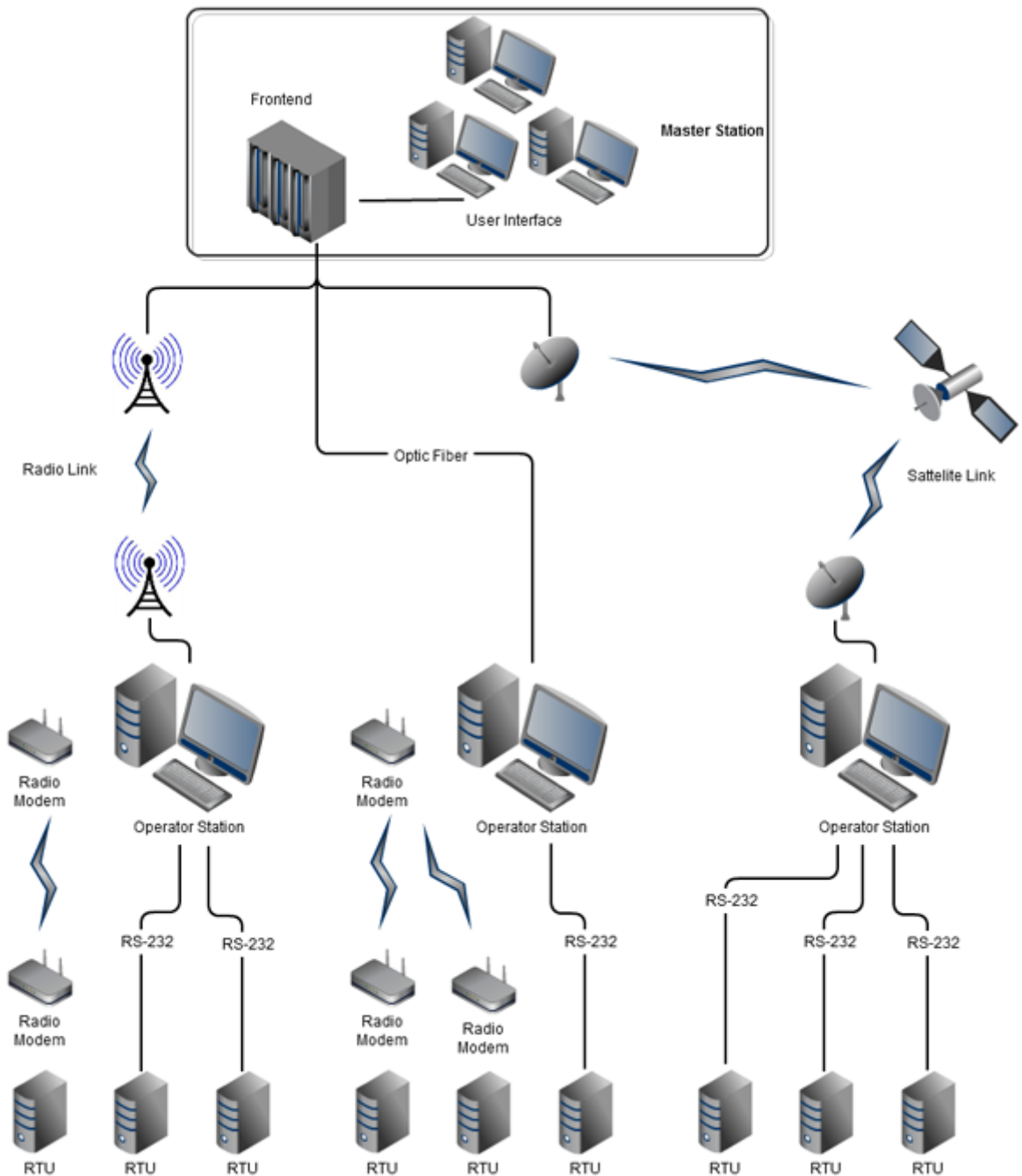


Figure 2.2: Typical SCADA system

A brief definition of each layer and its main functions it's presented below:

- **Application** - provides the interface to access the service implemented by the communications architecture;
- **Presentation** - defines the structure of the information between the systems, describes the type of data, the acceptable range of values and provides encryption and decryption of data;

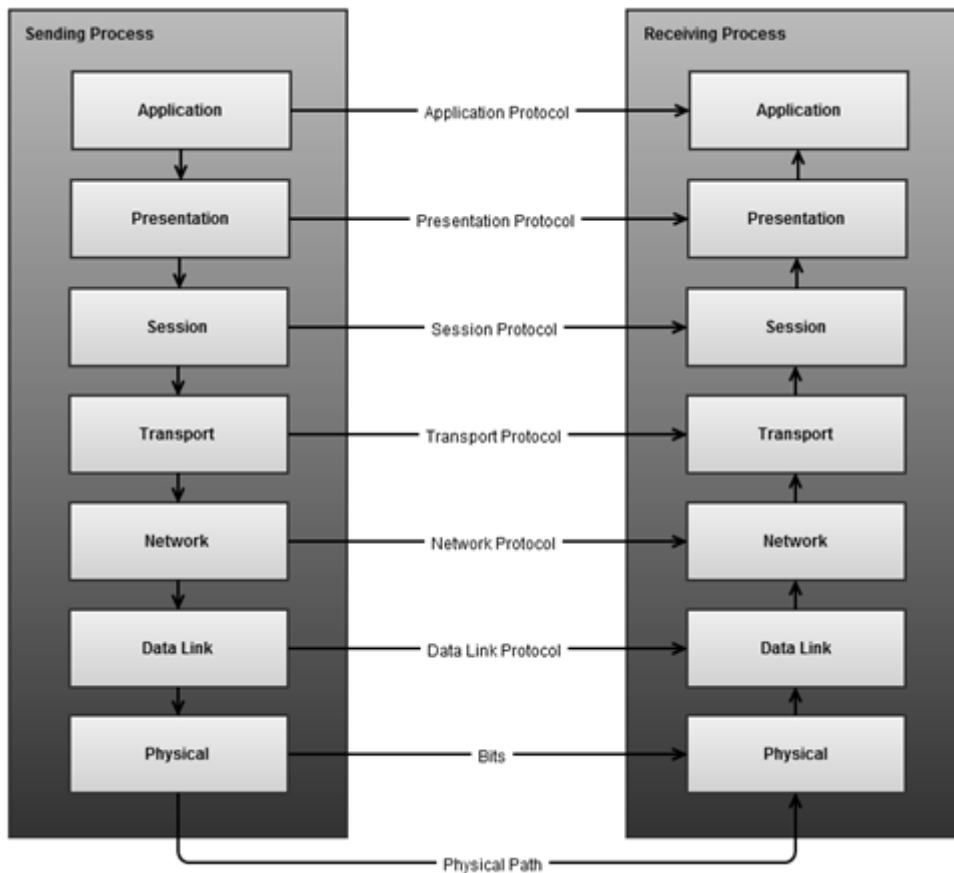


Figure 2.3: OSI Model's Structure

- Session - establishes, manages and terminates connections between users and applications' sessions, ensuring the synchronization;
- Transport - assures that the packages are received in an orderly manner and without errors, providing reliable transport between users;
- Network - states the network topology and ensures the routing of data between the systems;
- Data Link - provides transmission with error detection between two systems by physical addressing;
- Physical - ensures a data path between two points in a physical environment, specifying the physical and electrical characteristics and the signal type.

2.2.2 Modbus Protocol

This protocol was developed by Modicon and it is one of the oldest protocols in industrial applications. It is easy to implement, largely accepted and provided by a wide range of suppliers [3]. Although it's not an official standard, through the years it has become an open, non-official

standard for industrial communications between RTUs and PLCs [8]. The Modbus protocol falls within the seven layer of the OSI model, the application layer, implementing communications between devices within a network in a Master/Slave architecture [3]. As in a traditional Master/Slave architecture, the master is the only device allowed to initiate the communication. The other devices answer to the message sent by the Master, sending the required information. This protocol uses an addressing scheme that allows broadcast messages or communication with specific Slaves. The same addressing scheme limits the scalability of the system due to the limited number of addresses [3]. Communicating with every Slave individually requires the reading of each register by the master, which may require a large bandwidth, becoming too demanding for applications with a large number of RTUs. Although its drawbacks and limitations in large applications, this protocol is highly efficient in simple networks where there's a small amount of information being transmitted [7]. To overcome these drawbacks it was created several new versions of this protocol that somehow bypass these problems (Modbus TCP / IP or Modbus Plus) [6]. Although there isn't an official standard that ensures consistency between suppliers this protocol is available in most devices and it's considered a standard between many companies.

2.2.3 DNP 3 Protocol

This protocol is a telecommunications standard for SCADA communications and was created by the Harris Controls Division (now owned by GE Energy) initially designed for electrical distribution systems, although now it can be found in many different applications like water distribution systems, security industries and oil and gas distribution [3]. It hasn't a large spread in Europe, where IEC 60870-5-101 is widely used, but it has a great impact and a high acceptance in North and South America, South Africa, Asia and Australia, being a very popular protocol and the principal IEC 60870-5 competitor. In these countries it is recognized by one of the major SCADA protocols, it is provided by a large number of suppliers and used in several applications in many different areas like electrical and water distribution systems [1]. DNP3 is also an open standard provided by the DNP3 User Group, available to everyone by a nominal fee. The protocol was created to achieve interoperability between systems in a distribution network and it defines communications between RTUs or any other remotely controlled system in a SCADA environment. The frame formats and message structures are based on the section 1 of IEC 870-5 (named Transmission Frame Formats) which was being developed at the time. Using the IEC 870-5 documents made DNP3 and IEC 60870-5-101 share some common ground, giving them some similar foundations in the data link layer, although they differ greatly in the higher levels of functionality and data objects [1]. Since the main objective to the development of this protocol was to be used in a SCADA system it was designed to transmit small amounts of data in a reliable manner, assuring that the packets arrive in a deterministic sequence. This could be achieved using TCP/IP using for example FTP, or SCP/SSH or even HTTP [9]. All of these could guarantee communications with the needed features but not in a way that would be completely suitable for SCADA control, since it usually transmits small amounts of data and it needs a slow delay, the overhead in other non SCADA oriented protocols can be excessive [9]. DNP3 supports several topologies: multiple

masters, multiple slaves, peer to peer and hierarchical structures with intermediate data concentrators. It can operate in different modes either polled or quiescent [1]. Quiescent mode provides better efficiency for the system's communications since it allows the slaves to initiate communications. This allows the system to be silent and only communicate when there are changes. Although there is always a background polling system that makes the master aware if any of the slaves lost the connection. Despite the slaves being able to initiate communications they can't request data or issue commands, making the designation "slave" still suitable in this case since those abilities are restricted to the master. Besides these features DNP3 provides broadcast messages, a large number of devices addresses by single link (65000), message confirmation, time controlled events and synchronization and optimum error control (dividing the messages into frames) [7]. The DNP3 was created to achieve several objectives providing it with numerous attractive features. The data link and application layer use the frame format defined by the IEC 870-5 which allows confirmation messages providing data integrity [10]. The implementation of an object based application layer with an adapting structure allows a wide range of applications while maintaining interoperability [10]. It provides different modes of operation (polled or quiescent) making it suitable for different applications [1]. As a layered protocol can have a wide range of applications over local and wide area networks alike. It was developed to maintain flexibility using a minimum of overhead possible, ensuring a high efficiency [1]. It is a non-proprietary evolving standard and the full specification of the protocol is available to any person [10].

2.2.4 IEC 60870-5 Standard

IEC 60870-5 was completed in 2000 and intended to define a set of open standards for the transmission of information and telemetry data in SCADA environments. Although it was initially focused to electrical applications (it has data objects specifically for those) it can be applied to any general SCADA system in any industry [7, 9]. The standard describes detailed functional operation for controlling equipments in a SCADA system with processes geographically dispersed. The standards are divided in six parts with a number of companion standards. Each part is divided in several sections that were sequentially published through the years. Part 5 is divided in 5 sections that provide the core specification for the transmission protocols. The companions add specific information about the field of operation defining different information objects applied to different purposes [1]. The IEC 60870 structure detailing transmission protocols, is pictured in 2.4.

IEC 60870-5-101 companion has the most meaning when IEC 60870 is mentioned in the context of SCADA environments. This document defines the data objects in the application level essential to SCADA systems, completing the full definition of a complete working SCADA protocol. This companion was able to finally provide data objects and the necessary application level functions to define a full working transmission protocol. It defines general communications with RTUs and data types specialized to electrical applications, although they are generic enough to be applied to other SCADA applications [11, 12, 1]. This companion provides link and ASDU addresses for end station classification, data is classified into different information objects with a specific address and can be transferred with different mechanisms depending on its priority. It

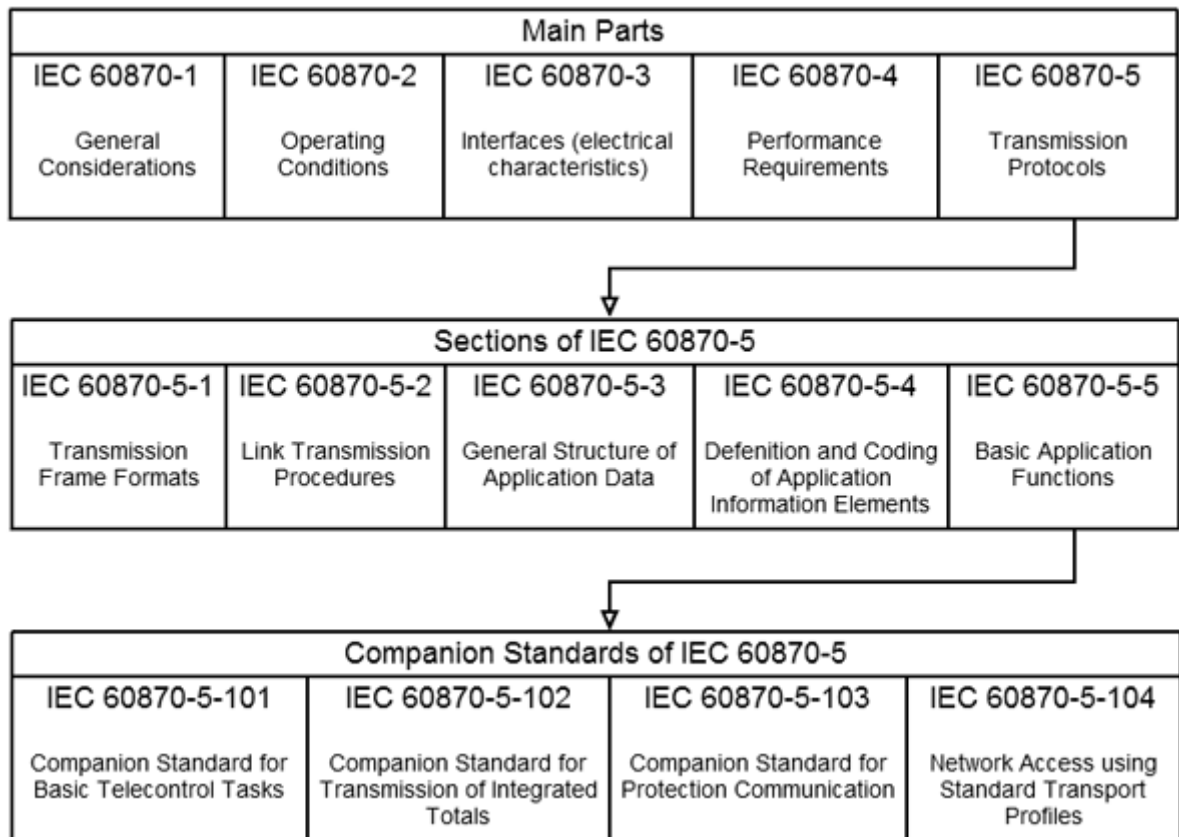


Figure 2.4: IEC 60870 Structure

also provides cyclic and spontaneous data updating schemes, time synchronization and file transfer functionalities. It supports multidrop and point-to-point communication links, with either balanced or unbalanced communication [10]. Balanced scheme is limited to point-to-point links and needs collision avoidance, making the system more complex, although it provides a better communications system's efficiency. On the other hand, unbalanced communications simplifies the system and doesn't need collision avoidance but only the master can send primary frames. Since balanced communications are exclusively for point-to-point links, in a multidrop topology the master must implement a cyclic polling scheme to interrogate the slave stations. The network access companion IEC 60870-5-104, it's also of particular importance. In this document it is defined how the part 5's messages are transported over networks. It refers to the usage of TCP/IP for the transport of protocols [13]. As expected this standard defines very different physical and data transport mechanisms but leaving the data objects of the higher layers unchanged. IEC 60870-5-104 and 101 aren't completely independent from each other. In the lower layers, while 101 defines every aspect of the protocol from the application layer to the physical layer, 104 uses existing functionalities in TCP/IP transport and network protocol for the message transport. In the application layer though, the two standards are completely equivalent, 104 uses the same functions defined in 101 [1]. It is important to detail some of the standard's specifications relevant to this project.

The two most important features that the Java application has to deal with are the frame structure and the link initialization. The frame structure is important because the application has to detect and process the standard's frame structure. It is important to know the link initialization process because the flow of the messages is essential to understand the tests and the logs.

2.2.4.1 IEC 60870-5 Standard Structure

There are two forms of frame formats, one with fixed length and another with variable length. The fixed length frame carries no user data and is used only for acknowledgments and data link control commands [1]. There is also a single control character used only for acknowledgment [1] but it is not used in the company's standard implementation. The picture below shows the overall frame construction of each type of frame at the octet level. The first octet is shown at the top, and following octets are shown below. The L field indicates the length of the frame from C to the end of the link user data. C is the control field and is used to identify the type of frame. A is the address field and contains the link address of the secondary station.

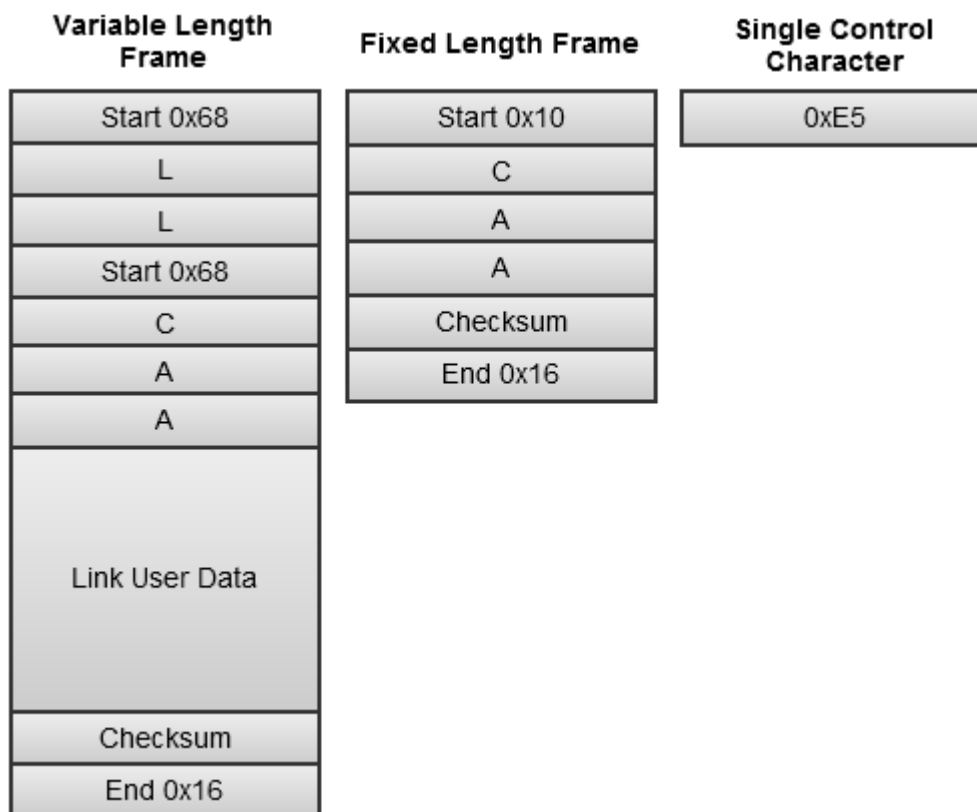


Figure 2.5: IEC 60870 frame types

The link user data is composed by one ASDU. The ASDU has two main sections, the data unit identifier and the data itself. The data unit identifier defines the specific type of data and provides

addresses to identify the data's identity. The data is composed by one or more information objects. Picture 2.6 details the ASDU structure.

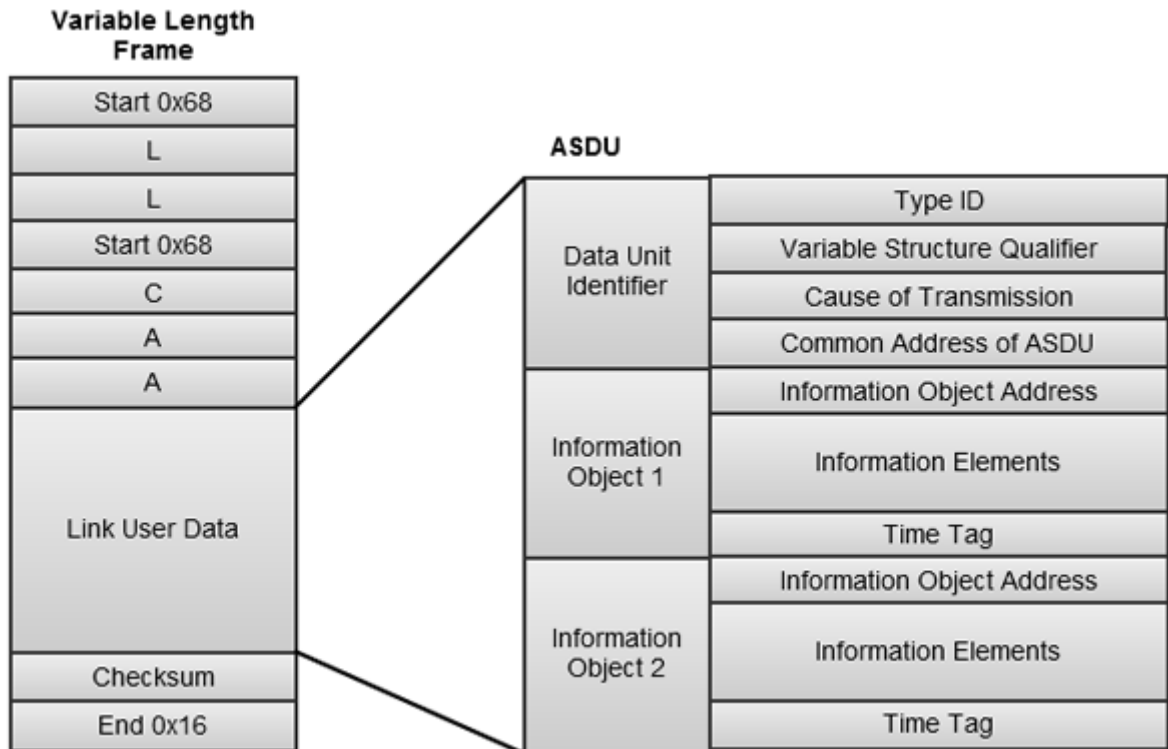


Figure 2.6: IEC 60870 link user data detail

2.2.4.2 IEC 60870-5 Message Sequence

The standard defines two main phases of message exchange: the link initialization and transmission procedures [1]. The information below is relevant for the company's implementation which is an unbalanced multi-point application. Link initialization is a service carried out when a station is offline and becomes available again. The master station periodically sends link status request functions until the slave station responds with the link status. For unbalanced mode the master starts by sending status request until the status of link is received, then the master sends the link reset and the link becomes active when the master receives an acknowledgement, the slave then generates a station initialization complete event [1]. After the initialization the controlling station will cyclically poll each station for any available class 2 user data. The secondary station will return any class 2 user data and will also indicate if there is any class 1 data available. The message sequence is presented in picture 2.7.

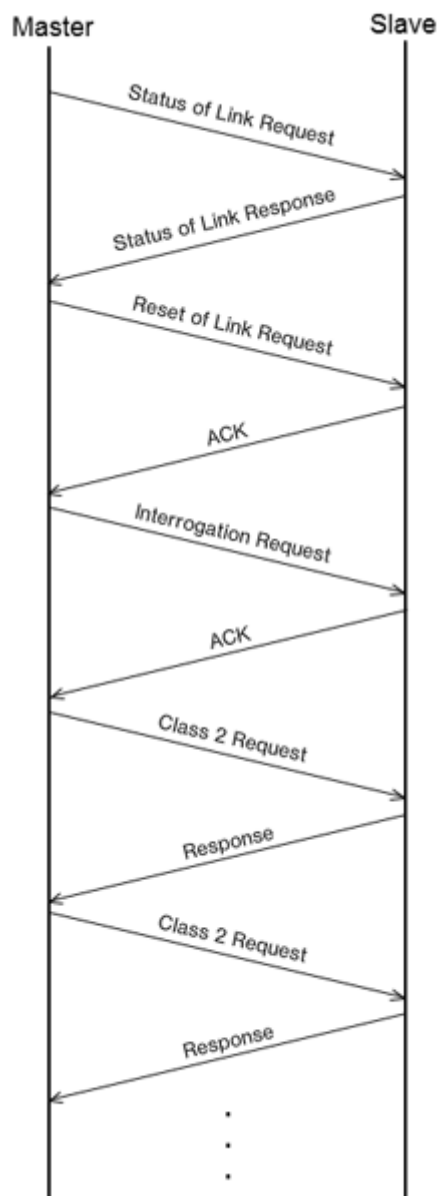


Figure 2.7: Message sequence diagram

2.2.5 Protocol Comparison

While IEC 60870 and DNP 3 are standardized (by IEC and DNP user's group respectively) Modbus is a proprietary protocol with no technical committee to guarantee interoperability between devices from different suppliers and create standards to implement new functionalities. Making two devices work under Modbus can be difficult while using a well standardized protocol makes this job much easier. Also when the system needs an expansion the user may be bound to use the components of the same proprietary system or change a considerable amount of parts to change to another's manufacturer's protocol. IEC 60870 and DNP3 have technical committees that work to implement new functionalities and ensure interoperability achieving high compatibility between

devices from different manufacturers. Modbus has exclusively a Master/Slave topology, not supporting reports by exception. With DNP3 and IEC 60870 besides being possible to choose several different topologies, they support balanced communications and a quiescent mode in DNP3. This allows that only changes in the system are reported, greatly improving the communications efficiency and the communications channel usage by reducing the bandwidth [1]. Besides these characteristics DNP3 and IEC 60870 offer many features that aren't available in Modbus. Some of these are time synchronization, data objects and functions suitable for electrical SCADA systems, pooling report by exception, data classes and unsolicited responses [10]. Also DNP3 and IEC 60870 support event time stamping at the remote device. This feature is important when a device fails, being the information and the timestamp of the event recorded in the device even if the communications fail. This information can be collected later preventing data loss and allowing the proper identification of each event. Besides these drawbacks Modbus usually has a lower implementing cost, its simpler and suitable for simple serial networks with low amounts of information being exchanged [3]. Considering that both DNP3 and IEC 60870-5-101 have emerged from the same frame formats defined in IEC 60870-5-1 they have many similarities. Besides the functionalities referred before they support freeze and clear counters, file download and upload and high security data transmission [1]. Despite their similarities they still have several areas where there is a considerable difference between them. IEC 60870 uses both link and application addresses when DNP3 only uses link addresses. This characteristic gives IEC 60870-5-101 greater flexibility in its addressing system [10]. IEC 60870-5-101 also has a larger point address range and uses variable address lengths, using less bandwidth when only a small amount of addresses are needed. Both DNP3 and IEC 60870-5-101 support balanced communications, however in the second these are limited to point-to-point configurations. This might overcharge the bandwidth and may become unacceptable in multidrop configurations [1]. They also have several differences in application functions and data objects, DNP3 separates the functions and the data objects obtaining greater flexibility but perhaps at the cost of complexity. DNP3 has testing procedures and authorities and defined minimum implementation levels being one of the strong features for DNP3. However these features are fading since DNP3 is older and had an early lead in this area, IEC 60870-5-101 has been catching up for the past few years. In IEC 60870-5-101 data objects have no variations, the point address scheme is simpler and uses single byte ACK transmissions on data link layer, these characteristics between others make IEC 60870-5-101 simpler than DNP3. Although these features make the protocol operate in a simpler manner, they also require additional configuration which can complicate the system integration [1]. Another practical consideration is the location of the SCADA system. DNP3 is considerably more popular in America while IEC 60870 it's dominant in Europe. It can be more or less difficult to find support depending on the application's geographic location and industry type. Table D.10 summarizes the main differences between these three protocols.

Table 2.1: IEC 60870, DNP3 and Modbus comparison

Feature	IEC-60870	DNP3	Modbus
Organization	IEC TC 57 WG 03	DNP User Group	Modicon
Architecture	Three layer architecture	Four layer architecture. Also supports 7 layer TCP/IP	Application layer protocol
Standardization	IEC Standard	Open Industry Specification	No standardization
Topologies	Peer-to-peer and multidrop	Peer-to-peer, multiple slave and multiple master	Peer-to-peer
Time synchronization and time stamped events	Yes	Yes	No
Polled report by exception and unsolicited responses	Yes	Yes	No
Data classes	Yes	Yes	No
Complexity	Complex but simpler than DNP3	Most complex	Simplest
Implementing cost	Higher	Higher	Lower
Configuration parameters	Baud rate, device addresses, balanced/ unbalanced, frame length, size of link addresses, size of ASDU addresses and structure of point number	Baud rate, device addresses and fragment size	Baud rate, mode (ASCII or RTU) and parity mode
Dominant market	Europe and Australia	North and South America, Asia, Australia and South Africa	Worldwide

2.3 GPRS Technology

GPRS is a packet-switching technology that supports data transfers in GSM networks. It separates data into blocks with appropriate size, despite their type or content and transmits them. The packets are then grouped and pieced together at the receiver. Using packet switching allows the device to be connected and ready to send information without reserving the communications channel making it available simultaneously for several users, allowing an increase of bandwidth per user [14]. It also improves the efficiency of the radio spectrum because it only uses the network resources when there is a need to transfer data. GPRS was designed to coexist with the GSM public land mobile network and is able to provide higher data rates and support longer messages improving the capacities of GSM networks [15]. It extends Internet and X.25 networks to wireless cellular networks, allowing any application running with IP or X.25 protocols to operate over a GSM cellular connection [16, 15]. The main features of the GPRS modem used to test the implementation and its main alternatives are briefly explained next.

2.3.1 Siemens TC65

This modem will be used to test the implementation of this project. It is suitable for machine to machine communications and has a set of very interesting features that makes it a very attractive solution for remote control. Supports several standard interfaces like bus, audio, ADC, serial and even a SIM card interface and provides multislot GPRS class 12. It has a Java software development platform making it a suitable solution even for complex applications allowing memory allocation and processing capabilities. Java also supports secure data transmission with HTTPS, FTP, TCP, UDP, SMTP and a TCP/IP stack via AT commands. It also has a plug and play functionality being easy and quick to integrate [17].

2.3.2 Cinterion MC55i

Also suitable for machine to machine communications the MC55i is an interesting alternative to the TC65 providing many of the same features with reduced power consumption and extended temperature range. Similar to TC65 it supports TCP/IP stack access through AT commands, UDP, FTP, POP3, HTTP and SMTP. Besides this features it has some interesting drivers for Microsoft Windows and Windows Mobile and supports GPRS multislot class 10 [18].

2.3.3 iTegno 39XX

Specifically designed for industrial applications, this GPRS modem is a simple and adequate solution for telemetry and data control. It's built to provide robust and compact wireless communications, at a low price with an easy and simple integration. Provides multislot GPRS class 10, integrated TCP/IP and UDP/IP stacks and user interface through AT commands. It also supports an optional low transmit power mode [19].

2.3.4 Robustel GoRugged M1000

Developed for machine to machine communications, this modem offers state of the art GPRS connectivity. Transmits data over GSM or GPRS mobile networks and supports Modbus RTU slave protocol, has software selectable interfaces and supports a wide range of input voltages. It also has the functionality to use a set of AT commands to control the modem. Offers GPRS data transfer through a TCP/IP stack, it can directly convert serial data to SMS without using AT commands and supports remote configuration through SMS [18].

2.3.5 ABB RER601 and RER603

RER601 and RER603 are wireless gateways and are able to guarantee versatile and reliable communications. They have built-in secure VPN connection and GPRS communication and are able to provide a secure connection between the master station and substation. This solution connects to SCADA via the M2M gateway using the standard IEC 60870-5-104. It uses a secure VPN connection to any public GPRS network provider. These devices can also connect to other devices

using the IEC 60870-5-101 standard and can also convert the information between both 101 and 104 protocols. This solution can be used to automate distribution networks, remote substations or transformers.

2.4 SCADA Satellite Systems Providers

Communications in SCADA systems are usually implemented with optical fiber or microwave. However these methods are very vulnerable when major natural disasters occur or even too costly or difficult to implement in some locations. Satellite systems are the best option in these situations. They are robust against natural disasters and become cost effective where optic fiber isn't. Satellite systems become a reliable backup communication system improving the communication reliability [20]. The main data transmission services over satellite communications will be briefly described next.

2.4.1 Orbcomm

Orbcomm's satellite network focuses on machine to machine communications supporting satellite and cellular data transmission. They offer solutions for remote control, management and device tracking. Their services manage two way data communications, combined satellite and cellular communications, networks management and full control of remote applications. The network uses low Earth orbit satellites providing low delay communications and lower signal attenuation. With each link the satellites scan the available frequency bands and dynamically assign channels to users minimizing interference. The satellites are simple, reliable and have longer lifecycles lowering the cost to support the satellite system providing lower subscription prices [21, 22].

2.4.2 Thuraya

Thuraya supports voice and data communications worldwide excluding the polar regions, through an intelligent satellite network. The network is able to allocate resources dynamically minimizing signal congestion, varying the transmitted bandwidth to adapt demand. This makes the communications link more reliable and stable, reaching farther locations by increasing the satellite power and concentrating the capacity on specific beams. Satellite calls may also be switched in space, increasing the efficiency of the network and providing greater flexibility to the terrestrial network components [21, 23].

2.4.3 Iridium Communications

Iridium Communications Inc. uses a satellite communications system to offer real time and reliable voice and data communications. It guarantees global coverage through a constellation of 66 low Earth orbiting cross linked satellites. All of the satellites' spot beams overlap minimizing missed connections. The satellite constellation works similarly to a mesh network, each satellite

communicates with nearby satellites in adjacent orbits. The information received is passed automatically to all other satellites and transmitted in their footprint. Iridium's architecture provides increased reliability through the multiple routing paths, when a single link is down the system is able to detect it and find an alternative path. Also low orbit satellites provide a shorter transmission time and less signal loss, allowing end users to communicate with lower potency equipments and generally simpler [21, 24].

2.4.4 Inmarsat

Inmarsat provides reliable mobile satellite services with a wide range of solutions. It has three constellations with a total of 10 geosynchronous satellites. These manage to provide a global coverage of broadband voice and data communications and two way data connectivity for messaging and monitoring of fixed or mobile installations. They offer strong communications links guaranteeing great reliability with availability above 99.99%. The broadband services are able to support complex applications and simultaneous voice and data communications. It also provides VPN products and encryption standards, supports the latest IP and switched circuited services with easy integration with legacy applications and provides plug and play terminals for machine to machine communications [21, 25].

2.4.5 Globalstar

Globalstar provides mobile satellite voice and handset data services. It has 32 low earth orbiting satellites providing imperceptible voice delay and reduced signal loss. The system's software operates on the ground allowing faster and easier system maintenance and upgrade. The constellation covers 80% of the planet surface and different satellites can pick up the same connection. This makes the system redundant, guaranteeing that a link is kept even if the user moves outside the satellite's range. When two satellites establish a link with the same terrestrial gateway the transmission starts, guaranteeing that at least one link is kept. The gateway distributes the information to cellular networks or the internet. Maintaining the gateways and the system's software terrestrial make the services easy to manage and the system easy to upgrade and to expand [21, 26].

2.5 Overview

The presented protocols are well defined and all of them represent suitable solutions to implement remote telemetry in SCADA applications. According to the differences and features analyzed, IEC 60870 proves to be the most appropriate solution for the desired application. All of the satellite system's providers seem to offer interesting solutions to implement substation telecontrol. A comparison between solutions and prices applied to the desired application will be presented later on. All of the GPRS modems presented would fulfill the requirements for this project but Siemens TC65 presents several advantages for allowing the development of personalized Java applications and for being a cheaper choice. Also the company has some experience with these

modems and it is easier for the technicians to use something they already know than something entirely new.

Chapter 3

Specification of the adopted solution

This chapter describes the functional specification of the adopted solution, presenting its overall architecture. It identifies the functionality offered by the system to support the initially proposed objectives, presenting a formal description of the system through the use of various formal diagrams, namely, use cases, component, collaboration and class diagrams. It thus provides further details concerning the objectives of this dissertation and the functionality offered by the system. It presents the system's actors and main architectural components and how they interoperate to deliver the desired functionality and thus fulfill the proposed objectives.

3.1 Objectives and Functionalities

The main purpose for this dissertation is to explore alternative ways to implement telemetry and telecontrol of remote substations within an electric power distribution network. More specifically, it aims at implementing an automated system that uses GPRS modems to transfer telemetry data between SCADA Frontends and Substations using the IEC 60870 standard. The solution requires the development of software to be deployed in the modems to manage the transmission of the protocol's messages. Besides this solution the dissertation also briefly explores the possibility to telecontrol Substations using satellite systems. Going beyond the initially proposed objectives, a Web-based online remote monitoring and configuring application was developed to allow remote users to manage the modem's network. This Web application had not been foreseen as part of the initial objectives and introduces new interesting features to the basic system. The final solution that was obtained is of the type "plug-and-play". When one modem is connected to the Frontend the communication link is automatically established with the modem connected to the Substation. The software implemented in the modems should be able to communicate with the GPRS network equipment, configure the connection, manage the Packet Data Protocol (PDP) context and manage the data flow between the mobile network and the RS232 interface. When one modem is connected to the substation the application on the Frontend's side should be able to negotiate its own IP address and communicate to the Substation's modem. It is also expected that the applications run for several hours, days or even months on possibly remote and hardly accessible locations and it

is unacceptable to expect that the technician will have to go there often. So it is essential that the connection stays operational for a considerable amount of time without requiring maintenance operations. For this reason several extra functionalities were implemented to make the solution more robust, being able to recover from error conditions. The application's performance was also a critical aspect due to the Frontend's timeout. A standard IEC 60870 Frontend has a configured timeout to receive the Substation's responses. If the response takes longer than the timeout it will be discarded. Usually this timeout is considerably small so it is important that the application introduces the smallest possible amount of delay. Having this into consideration, several additional performance optimization features were experimented and introduced in the final solution to minimize the processing delay. The Web application, providing an online interface, offers several additional functionalities that go beyond the initial requirements. Its main purpose is to provide a tool to monitor the modems' activity and configure them remotely, without the need to install special software as a simple Web browser can be used. This application is also self-sufficient and it is capable of updating itself without any user input. It provides detailed information about all the modems and SIM cards registered in the system, their geographic location, data sent and received over time and supports the remote interaction and modification of the modem's configurations.

3.2 System's Architecture

The system has three main components: the substation, the command center and the online server. The substation's equipment records the telemetry data and gathers them in an industrial PC. The industrial PC connects to the communications channel and sends the data. The command center gathers the data from several substations and displays them in user interfaces. Since there can be many substations transmitting to the same master station, a Frontend manages the incoming information and sends the corresponding answers. The Substation and the Frontend are the main components and the solution's main purpose is to support the connection between them. As referred above, the online server did not appear in the initial design of the system and in fact it goes beyond the initially stated objectives. It was added to support the online monitoring interface and it stores, displays and manages the information coming from the modems. The REST interface manages the interactions with the modems, all relevant information is stored in the database and displayed in the online interface. The system's high level architecture is pictured in figure 3.1.

The Intelligent Electronic Devices (IEDs) gather all the data from the Substation devices. This information is transmitted to the Substation's industrial PC and transmitted to the serial interface. The modem receives the data and sends them to the mobile network. The command center's modem receives the data from the mobile network and sends it to the Frontend's serial interface. The Substation's data is then displayed in the SCADA interface and new responses and commands are issued from the Frontend back to the Substation using the same mechanism. Besides managing the connection between the Substation and Frontend the modems report important information to the online server. To do this the modems access and use different Web services implemented by the REST interface to communicate the required parameters. The REST interface receives these

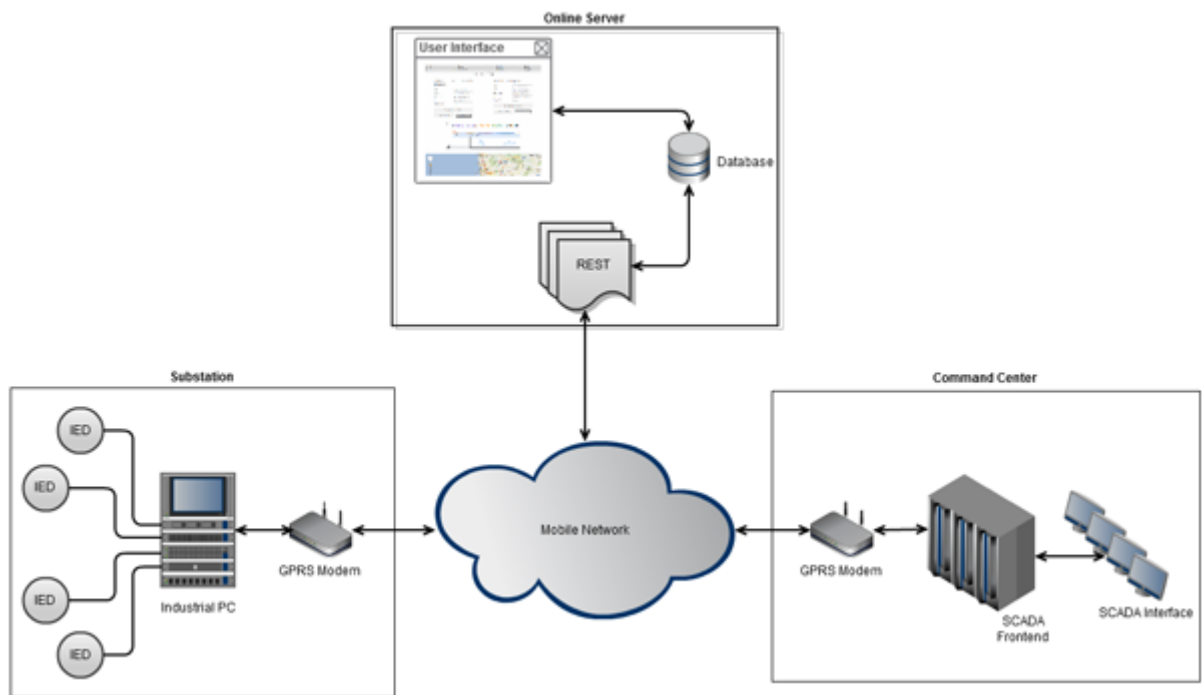


Figure 3.1: Overall system's architecture

parameters and stores them in the database. The online interface displays all information and allows to remotely configure the modem.

3.3 System's Specification

The system has three main actors: the technician, Frontend and Substation. The Substation and Frontend communicate with one modem and all the logic associated with that interaction managed by the Java application. The technician can either use the application's backend to interact with the modem or use the online interface. Each application has several different conceptual modules being similar to both client and server applications. The processing module is responsible for most processing operations, interprets the IEC 60870 frames and manages the interactions between all the modules. The serial and mobile interface modules create the connections with the serial port and mobile network and manage the information flow between them. The applications' backend is responsible for all functionalities related with the technician's interaction with the applications and the error recovery module implements the error recovery mechanisms. The Web interface implements all functionalities supported by the online monitoring interface and the REST interface manages the interactions between the modems and the online server and implements the Web services. The system's high level collaboration diagram is pictured in figure 3.2.

The application's main use cases are presented in the use case diagram pictured in figure 3.4 and the technician's interaction with the system is presented in figure 3.3.

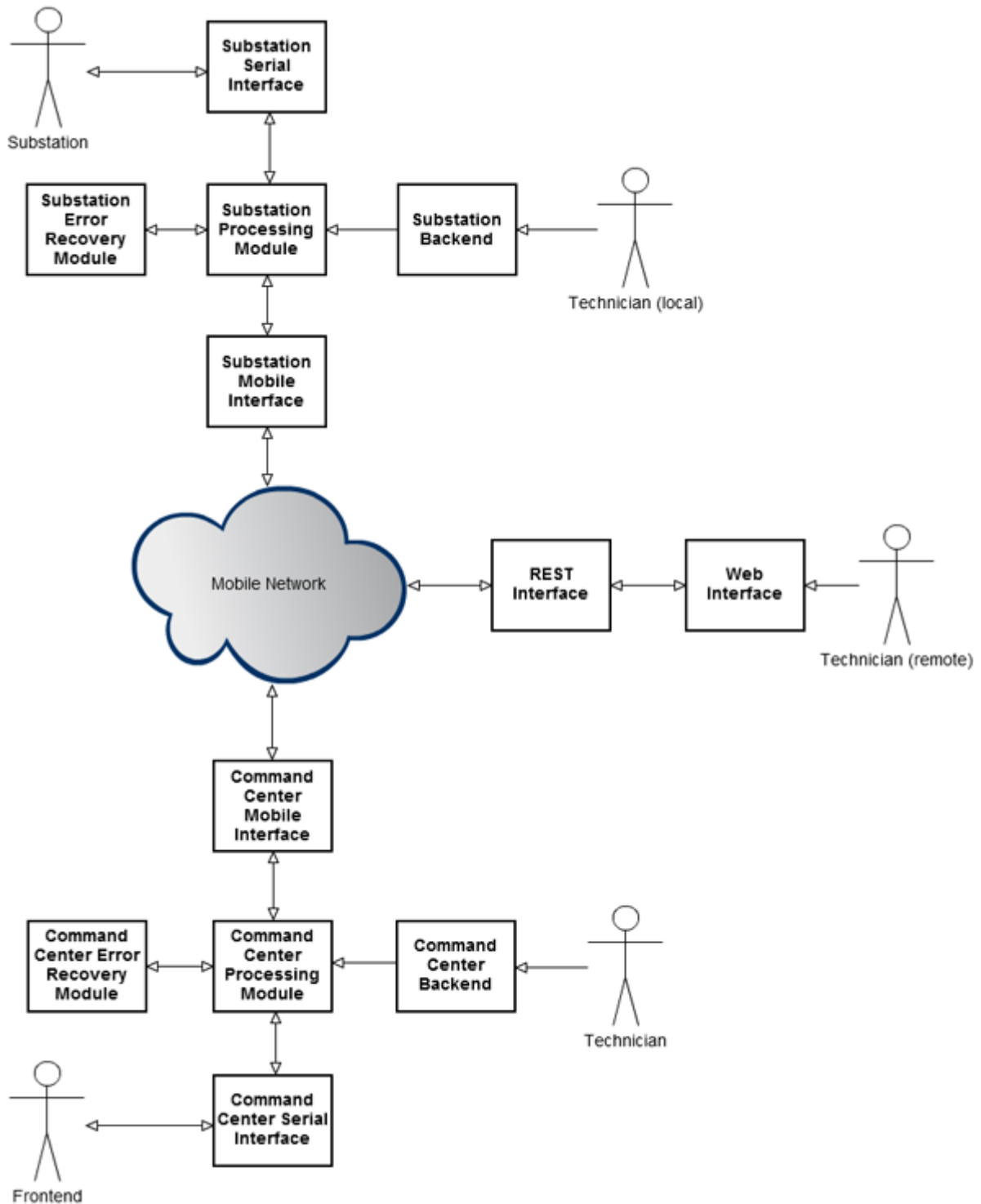


Figure 3.2: System's Collaboration Diagram

The detailed specification and course of events of each java application's use case is presented below.

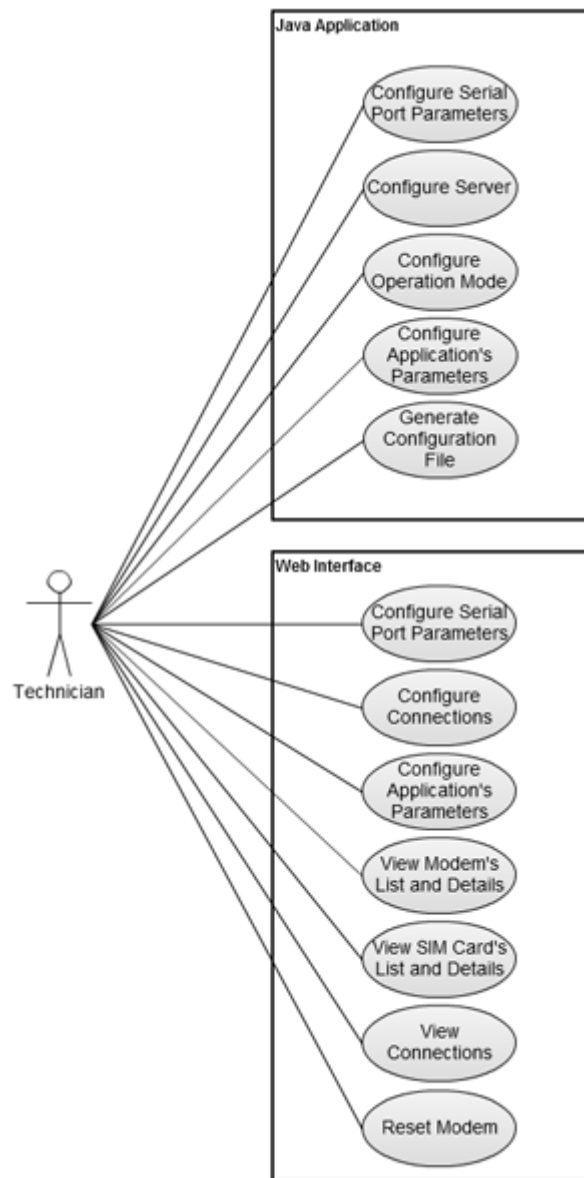


Figure 3.3: Technician's use case diagram

Table 3.1: Exchange data use case textual description

Use Case: Exchange data	
Code	UC01
Name	Exchange data
Description	The Substation or the Frontend sends the information to the serial port and the modem reads and sends it to the remote device using the mobile network. The remote modem reads the information from the mobile network and sends it to the serial port. The remote device can access the information through the serial port.
Actors	Frontend, Substation
Pre-conditions	Both modems must be connected.
Post-conditions	The data is sent by the device and received by the remote device.

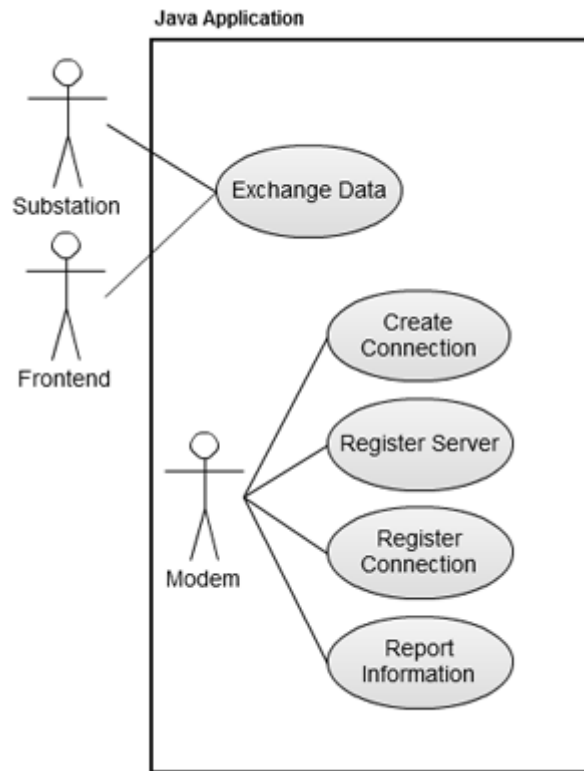


Figure 3.4: Application’s main use case diagram

Table 3.2: Exchange data course of events

Typical Course of Events			
Actor Action (Substation or Frontend)		System Response (Application)	
1	This use case begins when the Substation or the Frontend sends information to the serial port.	2	Reads the information from the serial port and verifies the frame’s integrity.
		3	Sends the information to the mobile network.
		4	Reads the information from the mobile network.
		5	Sends the information to the serial port.
6	Reads the information from the serial port.		

Table 3.3: Create connection use case textual description

Use Case: Create connection	
Code	UC02
Name	Create connection
Description	The Frontend's application creates the connection to the Substation's modem using the mobile network and the server's IP specified in the configurations.
Actors	Modem
Pre-conditions	None.
Post-conditions	The connection is created.

Table 3.4: Create connection course of events

Typical Course of Events			
Actor Action (Modem)		System Response (Application)	
1	This use case begins when the modem initiates and requests a connection to the remote device.	2	Reads the IP address defined in the configurations.
		3	Creates a connection to that IP address.
		4	The connection is successfully created.

Table 3.5: Register use case textual description

Use Case: Register	
Code	UC03
Name	Register
Description	The modem requests the registration web service to register itself on the online database. The parameters for that modem are sent in the request and the server returns that modem's configurations.
Actors	Modem
Pre-conditions	The connection with mobile network has to be established.
Post-conditions	The modem is registered and its information stored in the database.

Table 3.6: Register course of events

Typical Course of Events			
Actor Action (Modem)		System Response (Online Server)	
1	This use case begins when the modem sends a request to the registration web service.	2	Parses the parameters sent in the request.
		3	Stores the parameters in the request.
		4	Sends that modem's configurations.
5	Parses the configurations and loads them.		

Table 3.7: Register connection use case textual description

Use Case: Register connection	
Code	UC04
Name	Register connection
Description	The modem requests the connection web service to register the connection on the online database. The parameters relevant for that connection are sent in the request and stored in the database.
Actors	Modem
Pre-conditions	The connection with mobile network and remote device has to be established.
Post-conditions	The connection is registered.

Table 3.8: Register connection course of events

Typical Course of Events			
Actor Action (Modem)		System Response (Online Server)	
1	This use case begins when the modem sends a request to the connection registration web service.	2	Parses the parameters sent in the request.
		3	Stores the request's parameters in the database.
		4	Sends the confirmation.

Table 3.9: Report information use case textual description

Use Case: Report information	
Code	UC05
Name	Report information
Description	The modem regularly reports the modems activity from time to time accordingly with the configured time interval. To send the information the modem requests the statistics web service and sends the parameters in the URL. The service parses the parameters and returns any actions for that modem.
Actors	Modem
Pre-conditions	The connection with mobile network and remote device has to be established. The report functionality has to be enabled.
Post-conditions	The report data is received by the REST server.

Table 3.10: Report information course of events

Typical Course of Events			
Actor Action (Modem)		System Response (Online Server)	
1	This use case begins when the modem sends a request to the statistics web service.	2	Parses the parameters sent in the request.
		3	tores the parameters in the request.
		4	Sends any actions for that modem to execute.
5	Parses the parameters and executes the actions.		

Regarding the Java application requirements specified above, the preliminary class diagram for the application is presented in picture [3.5](#).

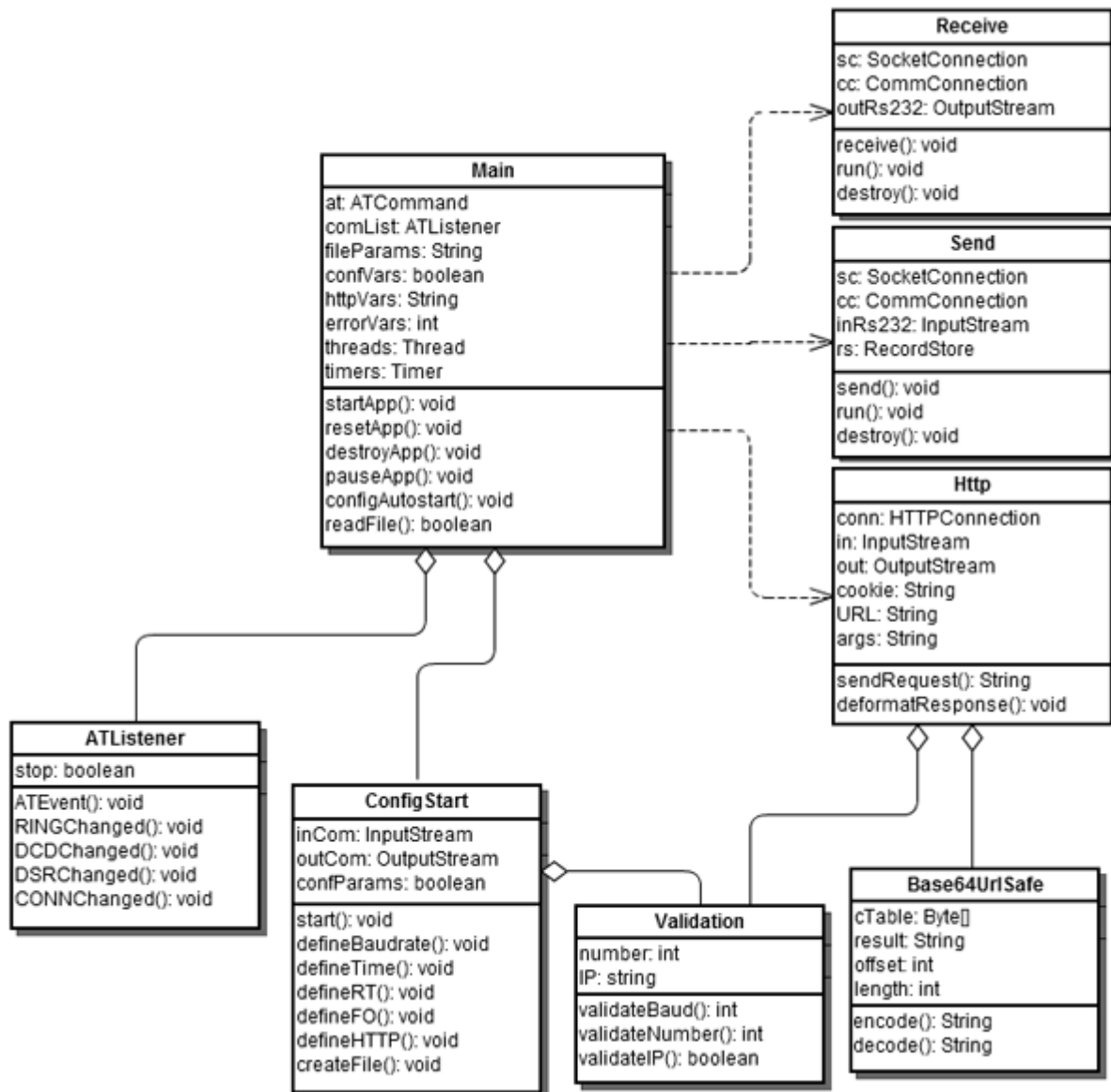


Figure 3.5: Java application class diagram

The diagram is simplified and does not include all the variables and methods implemented in the application. Also the variables `confVars`, `errorVars`, etc represent all the variables associated with configuration, error recovery, etc. If the diagram would have detailed all variables and methods it would become extremely large; accordingly, it was decided to present instead a higher level diagram. The `Main` class's `startApp` method runs when the modem initializes. This method initializes all the variables and uses the other classes according to the application's specification. The `Receive` class manages the data reception while the `Send` class manages the transmission. The HTTP requests are sent and interpreted by the `Http` class. The `ATListener` manages the responses of the module's AT parsers when AT commands are sent. The `ConfigStart` class creates the user interface, loads the new configuration parameters and creates configuration files. The `Validation`

class is used by both Main class and Http class and it is responsible for validating user input. The base64UrlSafe class encodes strings in base64 encoding making them appropriate to be used in the URL.

3.4 Impact in the Company

EDP is amongst the largest European operators in the energy sector, is one of the largest in the Iberian Peninsula and the largest industrial group in this sector in Portugal. It has over 564 power substations countrywide and an extensive power network throughout the whole country. Every substation is controlled and monitored remotely through an extensive telecommunications system. The power substation's telecontrol is usually supported using optical fiber or microwave links. The telecommunications system is able to support remote maintenance tasks in the power supply network reducing the electrical power unavailability time. The system specified within the context of this dissertation, whose functional specification has been described in this chapter, introduces a more flexible and less expensive alternative way to telecontrol power substations using GPRS modems. In fact, it provides an easier and fast solution to the existing infrastructure, namely when problems occur such as those that require to reinstate a broken fiber cable or to maintain the communications link during maintenance operations. Also it is the most adequate way to connect a Mobile Substation to the communications network. Mobile Substations are usually hard to connect because they require a whole PDH or SDH equipment and a new fiber node. This requires moving expensive equipment and moving fiber cables to the Mobile Substation that often is on the street and outside of the Substation premises. The online monitoring interface introduces interesting features for the departments in charge of overseeing the network. It allows the modem's configuration and monitoring from virtually everywhere and a powerful tool to configure and manage the whole modem network. Besides the main application this interface is applicable to any device with Internet connection. EDP also has an extensive medium voltage network with several devices, most of them modems Siemens TC65 also running different Java applications. The medium voltage network currently has 3126 Siemens TC65 modems installed and there is no registration tool or a way to find out which modem is where or to see any information about each modem. The online monitoring interface implements a way to monitor, configure and manage all of these 3126 modems and their connections completely automatically. The interface was also built in a way that any device with an Internet connection can use the services. This eases the scalability of the system making it easily applicable to future devices that may be included in the network.

Chapter 4

Development

This chapter describes the development work that was conducted within the course of this dissertation, describing the approaches adopted and the solutions devised to satisfy the proposed objectives. The work consisted on the specification and development of a client-server system, where the communication between the two remotely located components was established via GPRS modems. Both components were developed using JAVA technology. This chapter provides details concerning the main features of those components, the adopted workflow and the envisaged mode of operation. It is divided in five main sections: system overview, core functionalities, additional features; online monitoring interface; and performance considerations. After an overall description of the system, the second section describes how the main objectives for the project are fulfilled, by presenting the main features of the system and its operating mode. The main functionalities implemented on the online monitoring interface are detailed in the third section. The additional features that were integrated in the system, providing support for further functionality in relation to the basic initial requirements are described in section 4. Finally, section 5 discusses aspects related with the system performance, explaining specific procedures that were included to enhance the efficiency of the system and justifying choices made.

4.1 System Overview

The main purpose of the developed system is to automatically manage the exchange of SCADA telemetry data between a Power Substation and a Frontend module, using the IEC 60870 standard. To accomplish such objective, It was developed two applications were conceived and developed, namely a client application that was installed in the Frontend's modem, and a server application installed in the Substation's modem. Four main functional blocks, common to both applications, can be identified: initiation, reception, transmission and management. Their interaction is depicted in figure . The application starts by running the initiation functionalities, creating two new threads that will manage the reception and transmission of data. The management block is additional and goes beyond the initial project's objectives. It allows the modem's configuration and monitoring.

The initiation is responsible for variable initialization, modem configuration, serial port initialization, GPRS network registration, connection initialization and reception/transmission threads initiation. The reception thread manages the flow of information from the GPRS network to the serial port, while the transmission thread manages the flow of information from the serial port to the GPRS network. The management block configures several parameters available in an online database and reports the modem's statistics over time.

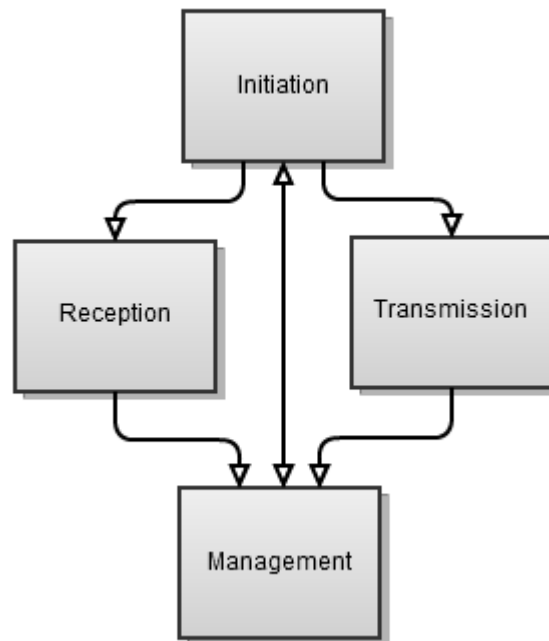


Figure 4.1: Java application's block diagram

4.2 Core Functionalities

The transmission and reception of information and everything related to their execution are the application's core functionalities, they fulfill the project's objectives and completely support the communication between the Substation and the Frontend. To achieve the requirements only the initiation, reception and transmission are necessary, the management block was implemented to support additional functionalities.

4.2.1 Initiation

The main initiation's main activities are variable initialization (timers, threads, GPRS profile declaration, etc.) and system configuration through the ATCommand interface. This interface allows the interaction with the AT command layer of the modem making it possible to issue AT commands within the normal flow of the Java application. AT commands allow the execution of various operations such as to configure the Autostart functionality, specifying if the application is

automatically started on power-up. This is an important functionality and it's always enabled in the application initiation, making the application more error resilient. If the application reaches an error condition or if an unexpected error occurs, the modem restarts and will start the application automatically. This will prevent the application to stop and endow it with the ability to correct itself if spontaneous and unexpected errors or connection losses occur. The error correction mechanisms are detailed further in the chapter 4.3.1.

The initiation is also creates the connection with the serial port and the mobile connection to the remote device. The serial port's connection uses the specified baud rate. Initially it was defined statically to the standard Substation's industrial PCs although later in the development it was developed several dynamical variable assignment mechanisms which define the baud rate dynamically. These mechanisms go beyond the project objectives and are detailed in the chapter 4.3.3. All of the functionalities detailed so far are shared by both server and client. This is not the case for the GPRS network connection. A socket connection is used to connect the client and the server using TCP. There were two options to do the connection, either UDP or TCP [27], the thought process and comparison between TCP and UDP is detailed in the performance considerations in chapter 4.5.6. The Java Micro Edition implements a socket connection using one device as server, waiting for connections, and other device as client, trying to connect to a specified server. At this point the server creates a server socket connection and blocks waiting for connections, while the client tries to open a socket connection to the defined server. The initial requirements for the system defined that the server's IP could be assigned statically, since the SIM cards owned by the company had fixed IP addresses. Regular SIM cards have dynamic IP allocation but the ones that the company provided had static IP addresses that they arranged with the operator, so the initial solution would work but in order to make the solution more flexible, the server's IP address was incorporated in the dynamical variable assignment mechanisms that were developed and detailed in the chapter 4.3.3. If both server and client have correct configurations the connection is made, otherwise the client will throw a connection not found exception, triggering an error condition making the modem reboot.

Once the socket connection is established between the two devices two new threads are launched to manage the reception and transmission of information. The main thread runs an infinite loop and only terminates the application if an error condition is met. The client's and server's application initiation high level flow chart is pictured below.

4.2.2 Reception

The reception thread manages the information received from the mobile network and sends it to the serial port interface. A loop keeps polling the socket input stream for new data. This loop is infinite unless an error condition is met. When new bytes are available for reading they are stored until there is no available data left. The stored data is then sent to Substation or Frontend through the serial port interface. In this thread the flow of information is mostly transparent since the error detection occurs in the transmission. There could be error detection both on the transmission and reception but by doing this the application would exchange delay and processing capabilities for

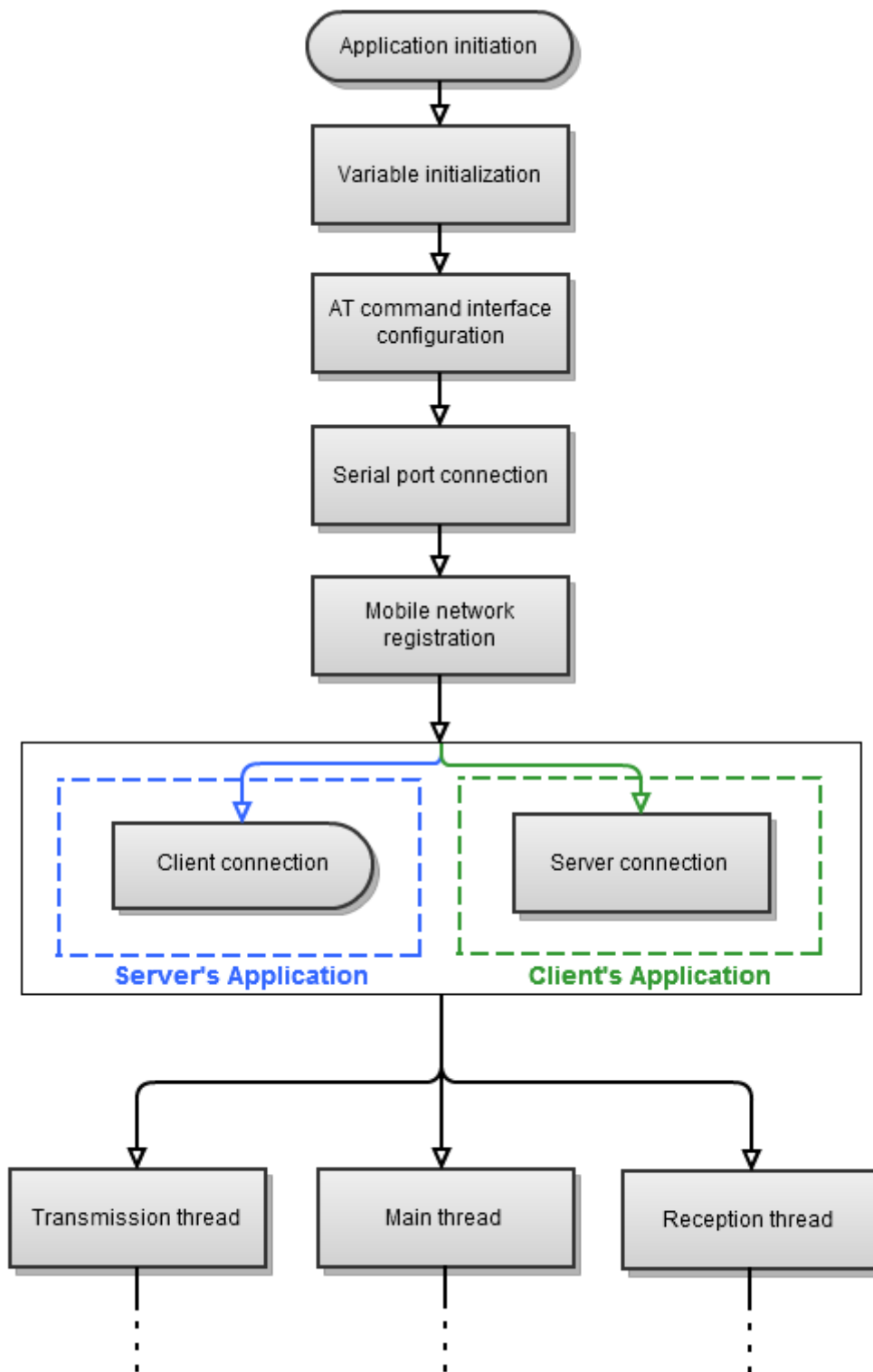


Figure 4.2: High level applications' initiation flow chart

no or little significant functionalities. Since the connection is made through TCP it ensures data ordering and reliability, being highly unlikely to receive information errors through the connection, so the optimization of processing delay justifies the lack of error verification. Also the Frontend already has error verification so there's no need to burden the application with exhaustive verification. Besides it is highly beneficial to do the error verification in the transmission thread instead of the reception thread since if occurs an error resources won't be spent in sending that information. Since the mobile network belongs to a private company there are always costs associated with sending information through that network. And since it is guaranteed that the information sent to the TCP connection is a correct frame it's very likely that a correct frame reaches the destination without compromising the processing delay excessively. The error correction mechanisms implemented in this thread are detailed in the chapter 4.3.1. The reception thread's high level flow chart is pictured in figure 4.3.

4.2.3 Transmission

The transmission thread manages the information received from the serial port and sends it to the TCP socket connection. This thread works similarly to the reception thread. Like the last, a loop polls the serial port input stream for new data infinitely unless an error condition occurs. When there are available bytes in the serial port they are compared with the IEC 60870 frames structure. Initially this comparison was done in a similar way with other solutions developed by the company in programs running different protocols, using simple comparisons [28]. For this solution it was developed a different, more efficient way using a state machine. This implementation and the comparison between the two are detailed in the performance considerations in chapter 4.5.5. If the data received from the serial port has a different structure than the IEC 60870, it's automatically discarded. Otherwise it's sent to the remote device through the TCP connection. The error correction mechanisms implemented in this thread are detailed in the chapter 4.3.1. The transmission thread's high level flow chart is pictured in figure 4.4.

4.3 Additional Features

Throughout the project it was implemented several additional functionalities. They include different variable assignment mechanisms, error correction mechanisms and different operation modes.

4.3.1 Error Correction Mechanisms

This application is meant to replace and to represent a suitable alternative to the regular communication technologies to transmit SCADA telemetry data between Frontends and Power Substations. Frontends are usually near the master station and therefore easily reachable, however the same doesn't happen with substations. The same maintenance department can cover distances to hundreds of kilometers and the technicians are expected to move to the locations very rarely. This application will be expected to run for several hours, days or even months on possibly remote and

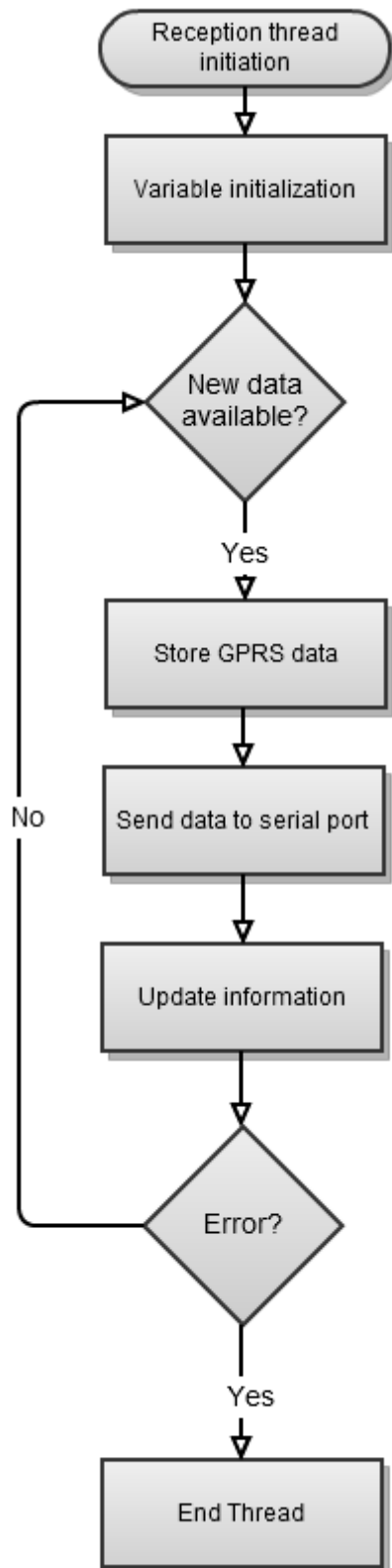


Figure 4.3: Reception thread's high level flowchart

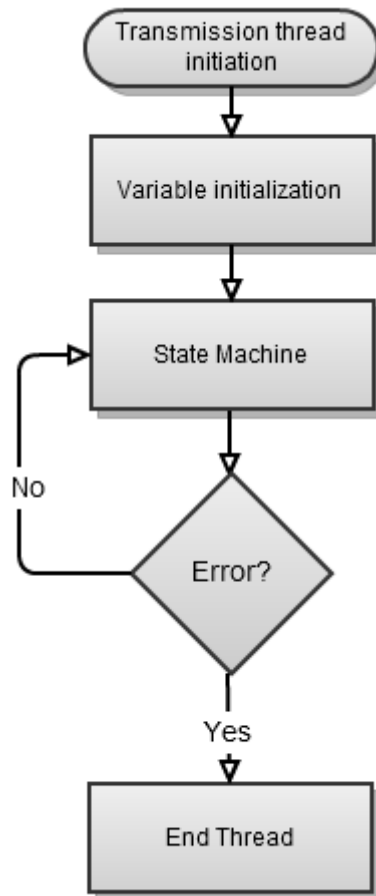


Figure 4.4: Transmission thread's high level flow chart

hardly accessible locations and it is unacceptable to expect that the technician will move there often. So it is essential that the modem will be able to run for a considerable amount of time without being the target of maintenance operations. During that time it is expectable that the modem or the application itself, encounters unexpected error situations like operator's network signal loss, connection failure, memory overflow, power failure, serial port connection problems, deadlocks or any other unanticipated error situations. It is important to foresee mechanisms to surpass, as far as possible, these error conditions.

4.3.1.1 Automatic Restart

Known as a common rule of thumb restarting the system solves many problems. This can solve illegal state conditions, memory overflow, unexpected runtime hangings or even connection problems. Through the AT commands interface it is possible to force the modem to shut itself down. It is also possible through the same interface to configure the Autostart functionality to make an application run automatically at start up. Joining the two and integrating the AT command interface

in the application it is possible to make the application restart within its natural flow. The Autostart functionality is enabled in every application initiation. This guarantees that the application will run the next time the modem starts, automatically restoring the application. This can prevent maintenance operations for example in electrical power failures, being the application restored as soon as the electrical power is available again. The structure of the program is prepared to handle events that disrupt the normal flow of instructions on main procedures that occur during the program's execution. When one of these events occurs the event is logged and the reset mechanism is activated. If the real time reporting of events is enabled, the error condition is also reported on the web interface, this interface is an extension to the initial structure, goes beyond the initial objectives and it is detailed in chapter 4.4. An AT command is sent to the modem's AT command layer to reset it. When receiving that command the system will try to shut itself down. It is still possible an unexpected event occurs when trying to issue this command or in its execution although highly unlikely. In this case there's no solution unless manual shutdown. Never the less this functionality is able to solve most of problems that may occur. Every time an exception is thrown when trying to run the core functionalities of the application a variable registers the occurrence of the error. After the initiation the application's main thread hangs on a loop which monitors this variable. This variable is shared between all the application's main threads. When an error is registered the application main thread ends terminating the application and the restart command is issued. This mechanism is implemented in the mobile network registration and communication, serial port's connection and communication and other vital parts of the application.

4.3.1.2 Communication's Timeout

When a problem occurs it is possible that it's not in the application itself but it's the communication's channel. This mechanism was implemented to allow the automatic detection of communication's problems. It is possible that there's a connection loss with the serial port or the mobile network. This kind of problem won't be detected because an exception isn't thrown and there's no way to actually be sure if the connection was lost or it's still viable. For example if for some reason the signal strength with the operator's network gets too low and eventually drops the connection, the application will still be sending information through the socket and it would never reach the destination. Even if the signal strength rises to normal values again the connection won't be restored and it is necessary to restart it. According to the IEC 60870 standard specifications relevant to this project, detailed in chapter 2.2.4, and the installation's protocol configurations, the frontend requests information once every a fixed interval [1] (one minute for the company's configurations [29]). So it is expectable that once in every minute a frame is sent and received. If this fails to happen it is safe to assume that there is a problem with either the reception or the transmission. If the problem is in the communication's channel itself (low signal strength, RS-232 cable disconnection, etc.) there's nothing the application can do at that level obviously. But it will be able to restore the connection as soon as the communication conditions are restored again. The reception and transmission thread each have a timer that schedule tasks for execution after the specified time. The timers are restarted every time a frame is sent, for the transmission timer,

or is received, for the reception timer. The timer's delay is specified using the dynamical variable assignment mechanisms specified in chapter 4.3.3. If the timer fires, it's very likely there is a problem with the communications and an error condition is met, triggering the automatic restart mechanism. By restarting the modem the connection with the serial port and the mobile network will be restarted as well, successfully restoring the communications. If there's still a problem external to the modem, the application will keep trying to connect until a correct connection is made.

4.3.1.3 Frame Offset

This mechanism is applied to the same context; to automatic surpass communication's problems. The mechanism counts the number of sent and received frames and compares the difference between the two to a specified maximum. This maximum is assigned with the dynamical variable assignment mechanisms specified in chapter 4.3.3. The communication's timer detailed in the previous chapter doesn't account for frames sent within the timer's delay but in an unbalanced manner. For example if the timer's delay is set to four minutes and within that time are sent four frames and only received one during let's say ten minutes, there is an offset between the transmission and reception frame count of thirty frames. According to the IEC 60870 specifications detailed in chapter XX, the standard accounts for a service with no reply [1] so the previous scenario was possible, this service however wasn't implemented by the company [28]. So we can assume two reasons for the previous scenario: either the reception delay is huge or there's a problem with the reception. If the delay is too large it will eventually cause communication's problems due to the Frontend's timeout. This problem would never be detected by the previous method because the timeout would never occur, since there's always a two way flux of information within the specified time. Another possible scenario that may happen is if the timer's delay is significantly higher than the polling interval, making the error detection excessively slow. For example if the pooling interval is one minute and the timer's delay is set to four hours, it will take at least four hours to detect the problem. When it could be a lot faster if the problem is either with the Substation's communications or the Frontend's, which is most likely than the both at the same time. When the problem occurs with just one of them the other keeps transmitting and it's possible to detect the problem faster by comparing the number of sent and received frames. Setting the maximum frame offset to let's say 50 frames, it would take 50 minutes to detect the problem instead of four hours and only 50 frames would be discarded instead of 240. This mechanism is implemented by introducing two counters in each one of the applications, one for the sent frames and one for the received ones. Every time a frame is sent or received the respective counter is incremented. If the offset between the two counters exceeds the specified maximum offset it will be interpreted as an error condition which will trigger the automatic reset functionality. There is an implementation detail that was taken into account when implementing this mechanism, when the counters are too large they are reinitialized to zero. This will prevent the overflow of the variables if the program runs for too long, as it's supposed to. The maximum value for an int in Java is 2147483647, if the variable is incremented again its value will be the minimum value for an int [30] (since the Java

Language Specification implements that the built-in integer operators do not indicate overflow or underflow in anyway, the results are specified by the language: `Integer.MAX_VALUE + 1 == Integer.MIN_VALUE` [30]), which is -2147483648. This initialization won't change the value of the offset at any time since they are reinitialized independently when a certain maximum is reached, being the difference always correct. This mechanism won't trigger for example if both the serial port and mobile network communications fail at the same time. Or if no information is exchanged at all since the offset will always be zero. These problems will still be detected by the communication's timers detailed in the previous chapter.

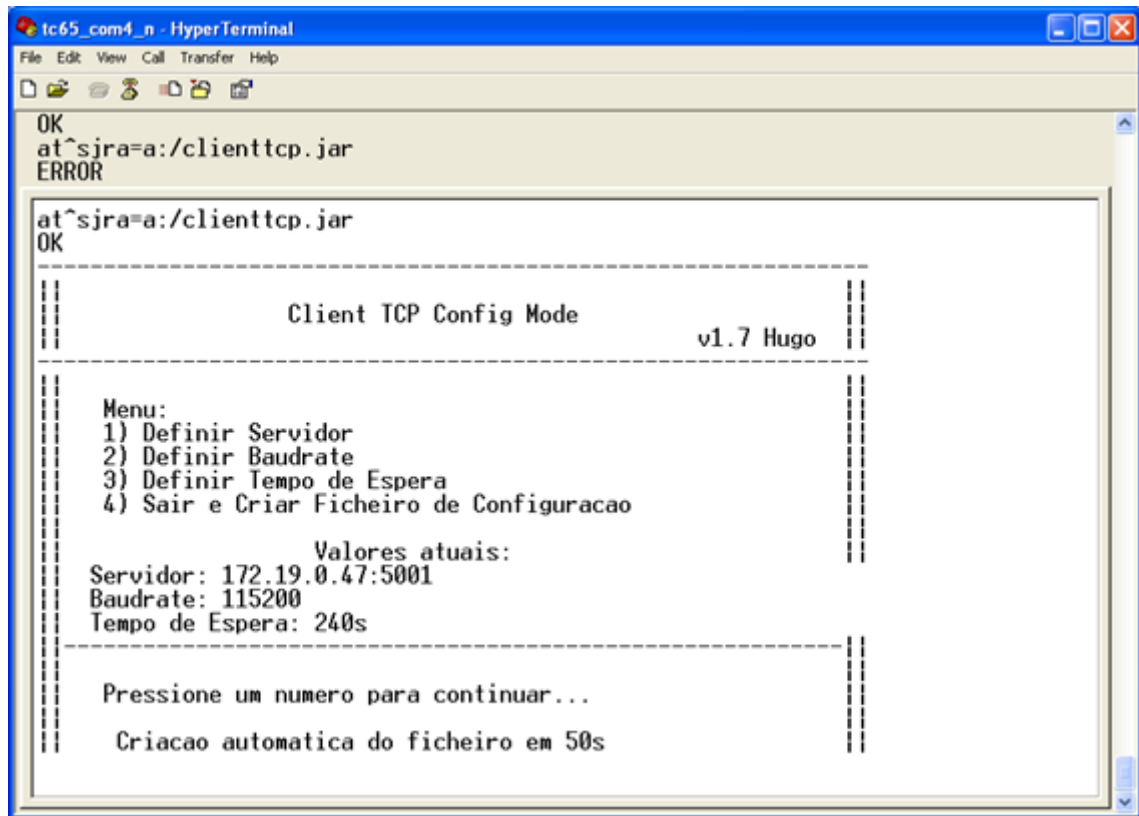
4.3.2 Operation Modes

It was implemented three different operation modes that run different functionalities. These modes aren't hardware dependent, they were implemented in the application and were designed to ease the technician's interaction with the modem. The available modes are: debugging, normal and configuration. Configuration mode allows the technician to manually assign values to the application's parameters. The modem has no user interface so it was implemented a way for the technician interact visually with the modem. This mode runs a user interface displaying a menu where the user can manually input the application's parameters. The interface runs through the serial port and it's displayed in any software capable of reading from the port. It can also generate a new configuration file, the contents and uses of the file are detailed in the chapter 4.3.3. This is important in case the file gets lost and the technician doesn't remember the file's structure. In that case it can simply input the values manually and a new configuration file will be generated. If the file is absent the configuration mode will automatically run allowing the technician to create a new file. The configuration mode will generate a new file with standard values when there's no user interaction. This was implemented in case the technician simply plugs in the modem, if no valid input is submitted it is assumed that the modem is connected to the Substation or Frontend. The application will restart after some time and run normal mode, resuming the main functions with the standard values. The configuration interface is pictured in figure 4.5.

Debugging mode was implemented to find the causes of possible error conditions. While in debugging mode the application displays every information message printed in the source code through the serial port. These messages will easily indicate why the modem encountered an error condition and can be used to surpass it. Normal mode implements the modem's regular functionalities and it's intended for the modem's normal execution.

4.3.3 Dynamical Variable Assignment Mechanisms

While implementing and designing the applications there were several parameters that had to be specified. The most important and most likely to be changed were: the server's IP address, the serial port's baudrate, number of bits per char and parity, the maximum frame offset, the communication's timeout, the reporting time rate, the application operating mode and whether it is relevant or not to use HTTP monitoring. Initially these parameters were assigned statically,



```
tc65_com4_n - HyperTerminal
File Edit View Call Transfer Help
OK
at^sjra=a:/clienttcp.jar
ERROR

at^sjra=a:/clienttcp.jar
OK

-----
Client TCP Config Mode                                v1.7 Hugo
-----

Menu:
1) Definir Servidor
2) Definir Baudrate
3) Definir Tempo de Espera
4) Sair e Criar Ficheiro de Configuracao

Valores atuais:
Servidor: 172.19.0.47:5001
Baudrate: 115200
Tempo de Espera: 240s

-----

Pressione um numero para continuar...

Criacao automatica do ficheiro em 50s
```

Figure 4.5: Configuration interface

although the applications would still work using static initialization it was far more useful if these parameters could be changed without actually altering the source code every time a change is needed. The variables are assigned using three different mechanisms: statically, by a configuration file or remote configuring. Initially the application uses the standard values declared in the source code. If a configuration file is present the application will load the values declared in the file. If the HTTP remote configuring is enabled the values will be requested to the online server.

4.3.3.1 Configuration File

To use this functionality a text file named “config.txt” has to be present in the application’s directory. This file needs a fixed structure or the file will be interpreted as corrupt. The structure validation prevents the assignment of erroneous values or the reading of other files that may exist in the application’s folder. The client application’s file structure is pictured below.

Every parameter assigned in the file is also validated. The application won’t accept a different IPv4 format for the server’s address, non-standardized baudrates, negative or non-numeric times or frame offsets, different modes than the implemented, etc. If any of the values has a non-accepted format the static value will be used instead or the remote value if the remote configuring functionality is enabled. If the file gets lost or if the technician doesn’t remember the structure

```
server=93.102.130.167:5001;  
baudrate=115200;  
waitingtime=240;  
reporttime=600;  
frameoffset=50;  
http=sim;  
mode=config;
```

Figure 4.6: Client application's file structure

a new file can be generated using the configuration mode, the implemented operation modes are detailed in the previous chapter.

4.3.3.2 Remote Configuring

If the HTTP remote configuring is enabled, the parameters can be assigned remotely using the online server. This functionality is integrated with the online monitoring and configuring interface detailed in chapter 4.4. The values are loaded with the initial modem online registration. By requesting the registration web service each modem gets a response with the corresponding remotely assigned parameters. The values suffer no validation since the input validation occurs on the web interface. The monitoring rate is the only parameter that can be changed during the application's execution. This functionality was applied to just this parameter because it's the only one that makes sense to be changed during the modem's normal execution. Since the other parameters are all used in the application's initiation changing them would mean the modem had to reboot for the changes to take effect. By restarting the modem the application automatically loads the new values anyway so there's no point in making those parameters dynamical during the application's execution. The remote configuring is detailed further in the online monitoring and configuring interface.

4.4 Online Monitoring Interface

In order to further control the connection and the solution's potential it was implemented an interface to monitor and remotely configure the modems. This interface goes beyond the initial dissertation's objectives and manages to implement additional interesting features to support the solution's scalability. Through this interface it is possible to view all the currently active modems in the network, their configurations and SIM cards, the geographically located connections and their details, graphically view the modems and connections sent and received frames throughout the time, the errors and main events, remotely configure the modems and remotely set up new connections or reconfigure old ones. All the information related to the modems and their connections are updated automatically without the need of any manual input. Since this is an online interface the technician can access all the modem's parameters, connections and configurations from virtually anywhere. Besides this, it is also possible to configure the modem with new parameters

and even restart it remotely. These features make the interface a powerful asset and considerably increase the solution's potential.

4.4.1 REST Interface Design

REST is a simple solution to manage interactions between independent systems. The interactions often exchange resources which are referred with a global identifier within the HTTP request. REST supports the scalability of devices and their interaction and it is flexible enough to guarantee that even if the devices change they can access the full capabilities of the interface. It fully explores the HTTP potential and supports interactions between different devices minimizing overhead. REST was used instead of other solutions (like SOAP) due to its flexibility to support personalized protocols that minimize overhead and optimize efficiency and delay, further considerations about the REST advantages are detailed in chapter 4.5.8. The implemented REST interface provides three web services: modem's registration, connection recording and statistics reporting. Each modem uses one of the services according the specific need of the program at that time. The following diagram pictures the program's web request time sequence since the beginning of the application.

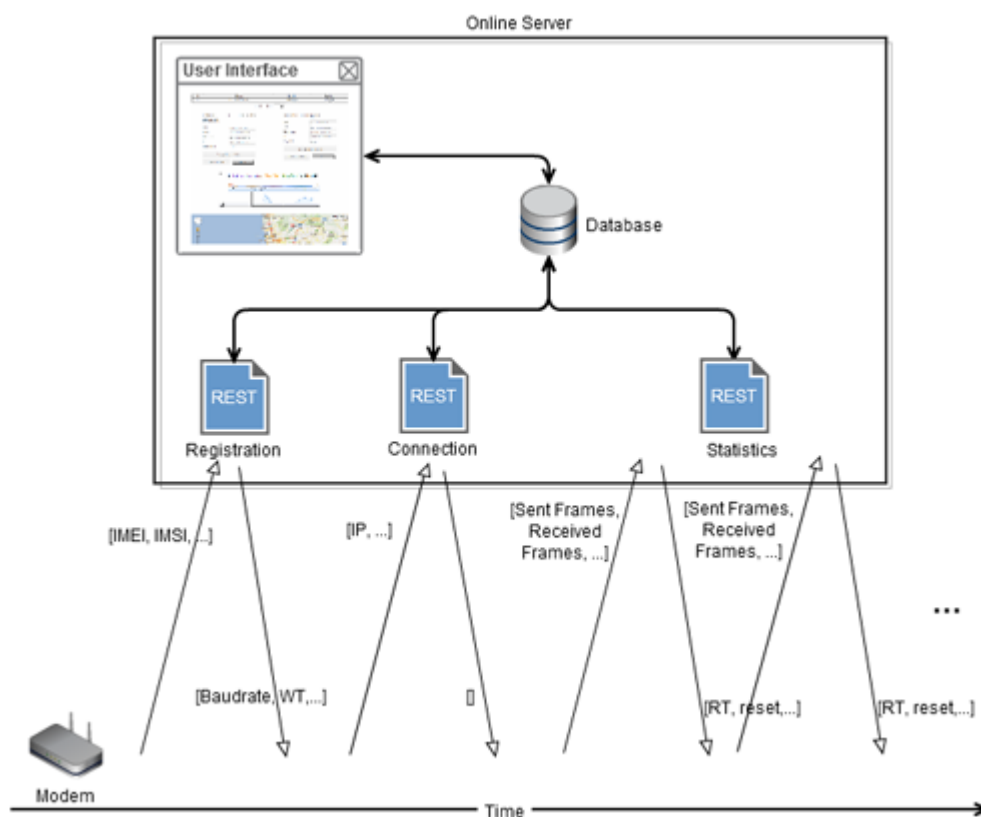


Figure 4.7: Application's web request time sequence

When the application starts it uses the registration service and transmits all the information about that modem. After storing this information in the database, the service answers with the configuration parameters to that specific modem. The application uses these parameters to initiate the serial port and mobile network connection. As soon as a connection between two modems is established the information about this connection is stored in the database using the connection service. The application regularly sends information to the online database using the statistics service. This service also supports the remote reset functionality, every time a modem reports any information it receives a response with any new parameters and whether a reset command was issued. The considerations about minimizing the HTTP requests to improve performance are detailed in chapter 4.5.7.

4.4.2 Interface Functionalities

This section presents and describes all the main functionalities supported by the online interface. It supports the display of information related to the modem's configurations, main events, connections, locations, SIMs and relation with the company's assets. It also presents images of the main visual interfaces.

4.4.2.1 Modems Listing

Every time a modem starts the application it automatically registers itself on the online database. A list of all the modems along with their IMEI, software version, configurations and geographical location is available on the interface. This listing is pictured in image 4.8.

IMEI	Info	Baudrate	Timeout de Espera	Offset das Tramas	Tag de Localização	Ver Detalhes
355632000891672	SIEMENS TC85 REVISION 03.000	115200	4 min	5	SPCBL-URT-UR	Ver Modem
354745033158462	SIEMENS TC85 REVISION 03.000	115200	4 min	5	0	Ver Modem
354745033158686	SIEMENS TC85 REVISION 03.000	115200	4 min	5	NSPLM-URT-UR	Ver Modem

Figure 4.8: Online modems listing

This table also includes the unique company's database tag of the substation where the respective modem is installed. The table can be ordered by each column and the user can cycle through the entries with the navigator below the table. Below this table the user can also see the geographic location of all the modems registered in the database. The geographic location of CC Alexandre Herculano is pictured in figure 4.9.

Also in this page the user can access each modem's details by pressing the last column. The details page will present the modem's IMEI, software version, baud rate, communication's timeout, frame offset, IMSI of the modem's SIM card, signal strength, IP address, the time of the last modem's and SIM's activation, connection ID, the modem's role, to which modem it is connected and the unique company's tag. In this page it is also available a graph with the modem's sent and

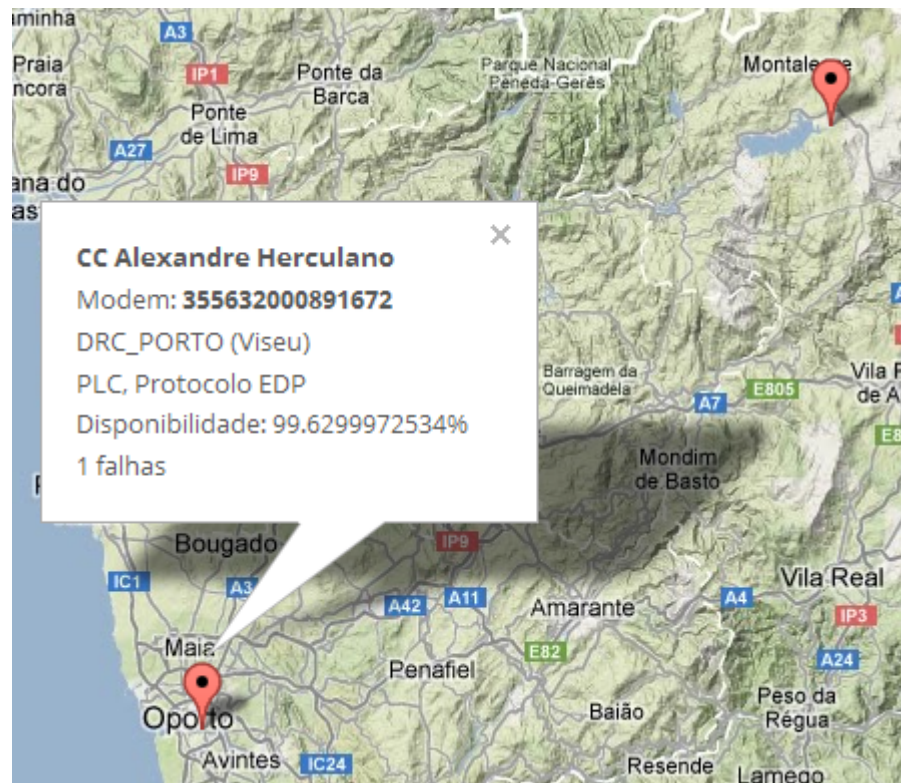


Figure 4.9: Modem's geographic location

received frames through the time. The modem's details interface is pictured in figure 4.10 and the sent and received frame graph in picture 4.11.

Modem [355632000891672]

IMEI	355632000891672	Ativação do Modem	2013-04-09 16:46:38
Versão de Software	SIEMENS TC65 REVISION 03.01	Ativação do SIM	2013-04-09 16:46:38
Baudrate	115200	ID da Ligação	14
Timeout de Espera	4 min	Função	Cliente
Frame Offset	5	Ligação com	354745033158686
IMSI	268032200092972	Tag de Localização	SPCBL-URT-UR
Força do Sinal	30.99		
IP	93.102.203.165:1025		

Figure 4.10: Modem's details

All of this information is inserted and updated automatically. Every time the modem is connected to a power supply it automatically sends all of the parameters and registers the number of sent and received frames from time to time, according to the monitoring timer. The modem's location is also updated automatically, the strategies to implement the geographic location detection are presented in the performance considerations in chapter 4.5.9. If one of the modems is installed

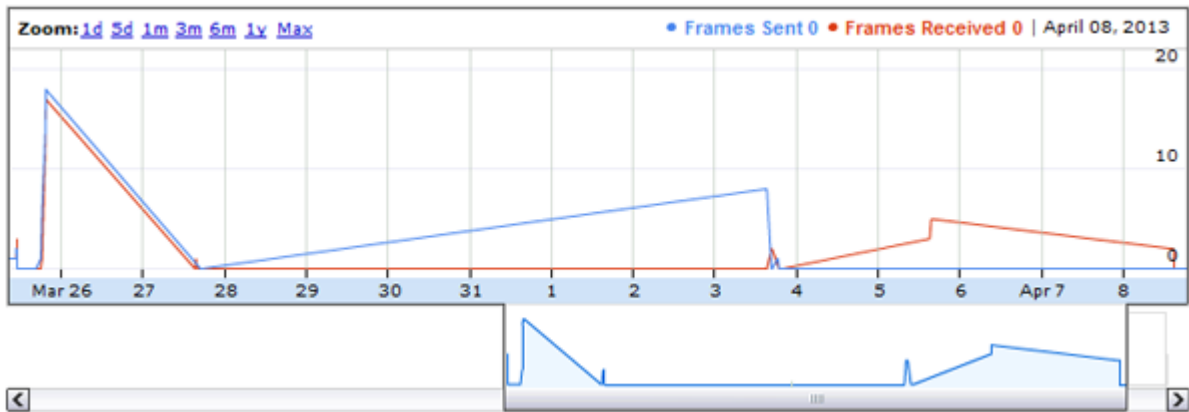


Figure 4.11: Modem’s sent and received frames graph

in a different location it will update all of the parameters automatically on start up.

4.4.2.2 SIMs’ Listing

When a modem automatically registers itself it also registers the associated SIM card at that time. The user can also consult a list with all the SIMs registered in the interface. In this list it is available each SIM card’s IMSI, the last signal strength recorded, the last IP address, the modem’s IMEI where the SIM card is installed and the last activated time. This information is pictured in figure 4.12.

IMSI	Signal	IP	Modem	Activated
268031104994776	30.99	93.102.130.167:5001		2013-03-21 17:10:36
268030202784499	22.99	93.102.141.186:1024		2013-03-27 18:59:11
268032200092971	31.99	93.102.202.246:5001	354745033158686	2013-04-09 16:45:37
268032200092972	30.99	93.102.203.165:1025	355632000891672	2013-04-09 16:46:38

Figure 4.12: Online SIM’s listing

The user can order the table’s entries by each column and navigate through the results using the navigator below the table.

4.4.2.3 Configurations

The user can also see the configurations for each modem. On start up each modem registers itself and the server returns a response with the predefined configurations for that modem. If there aren’t any configurations predefined the server records the current configurations. If there are already saved configurations in the database the Java application will parse the parameters and load them on the initiation. Considering this the technician can remotely configure the modem by simply editing the database parameters, the modem will automatically load them the next time it initiates.

And since the interface is available online one can configure every single modem registered in the network from virtually anywhere. The listing displays which modem has that configuration, the server's IP address, baud rate, communication's timeout, frame offset, monitoring time and whether the configuration is active or not. As detailed in chapter 4.3.1, the communication's timeout and frame offset are embedded in the error correction mechanisms. The monitoring time is the interval from how long each modem reports information, if it is for example 5 minutes that modem will update the information every 5 minutes. The active column informs whether the configuration is already active or not. The configurations listing is pictured in figure 4.13.

IMEI	IP do Servidor	Baudrate	Tempo de Espera	Offset das Tramas	Tempo de Monitorização	Ativo	Alterar
354745033158686	Funciona como servidor	115200	25 min	50	3 min	Sim	Editar
354745033158462	172.19.0.48:5001	115200	4 min	5	4 min	Sim	Editar
355632000891672	93.102.202.246:5001	115200	25 min	50	5 min	Sim	Editar

Figure 4.13: Online configuration's listing

The user can order the table's entries by each column and navigate through the results using the navigator below the table. By pressing the last column the user can change that configuration. The interface to modify the parameters is pictured in figure 4.14.

Alterar Configuração do Modem [354745033158462]

IP . . . :

Baudrate ▼

Timeout de Espera ▼

Tempo de Monitorização ▼

Frame Offset

Figure 4.14: Modem's configuration interface

The fields are initiated with the current modem's configurations and the user can change any one of them to the desired specifications. The baudrate is limited to the serial port's supported values and the times can be inputted in minutes, seconds or milliseconds.

4.4.2.4 Active Connections

Every time a connection is established it's also recorded in the online database. In the interface it is possible to see the list of all the existing and active connections. If the modem's location is

identified it is also automatically displayed the substation's and frontend's names. The connections listing is pictured in figure 4.15.

ID	Cliente	Servidor	Detalhes
14	CC Alexandre Herculano	SE Morgade	Ver Ligação

⏪ ⏩ 1/1 ⏪ ⏩ 10 ▼

Figure 4.15: Active connections listing

By pressing details the user can see that connection's details and a graph displaying the sent and received frames by both the client and the server. It is also possible to see the connection's geographic locations. This interface is pictured in figure 4.16 and 4.17.

ID	Cliente	Servidor	Detalhes
14	CC Alexandre Herculano	SE Morgade	Ver Ligação

⏪ ⏩ 1/1 ⏪ ⏩ 10 ▼

Cliente - CC Alexandre Herculano

IMEI: 355632000891672

Início: 2013-04-09 16:46:44

Último Envio: 2013-04-09 16:57:10

IP: 93.102.203.165:1025

Força do Sinal: 30.99

Ver detalhes do Modem

Reset do Modem Pronto

Servidor - SE Morgade

IMEI: 354745033158686

Início: 2013-04-09 16:46:44

Último Envio: 2013-04-09 16:57:10

IP: 93.102.202.246:5001

Força do Sinal: 31.99

Ver detalhes do Modem

Reset do Modem Pronto

Zoom: 1d 3d 1m 3m 6m 1y Max | April 03, 2013

Legend: Server Tx (blue), Server Rx (red), Client Tx (orange), Client Rx (green)

Figure 4.16: Active connections details

In this interface it is also possible to remotely reset each modem in the connection. If the user sends the reset command the interface will also inform when the reset was executed. This action state transition interface is pictured in figure 4.18.

subsubsectionPredefined Connections

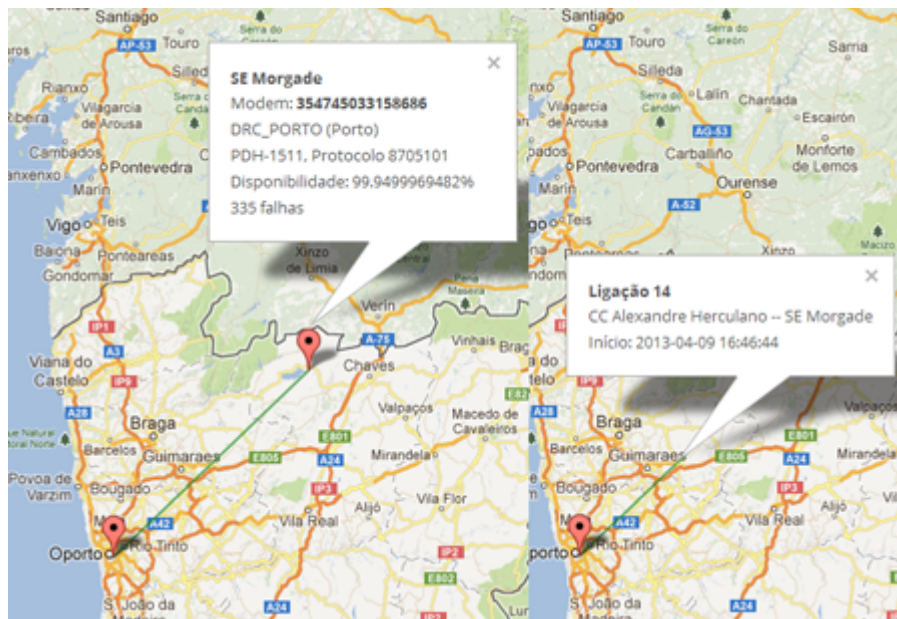


Figure 4.17: Active connections geographic location

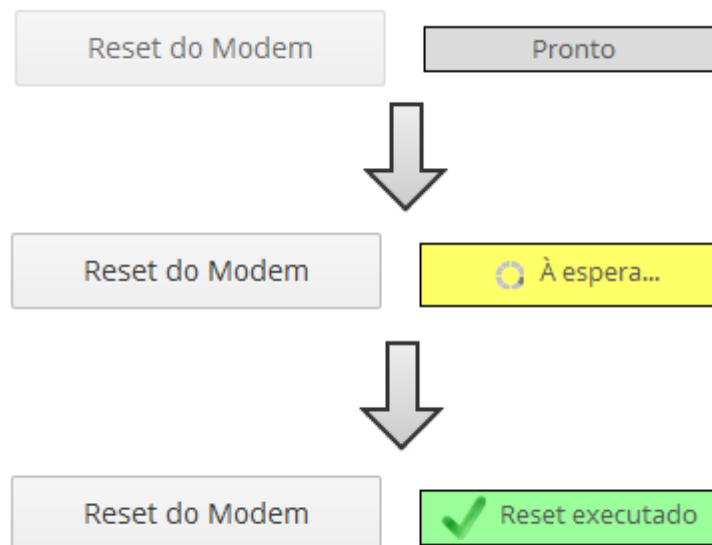


Figure 4.18: Active connections reset states

The user can also create new connections or change existing ones. By defining a new connection the client device when initiating, loads the predefined server’s IP address and attempts a connection. This feature is exceptionally useful when using SIM cards with dynamic IP addresses. Before this functionality was implemented it was necessary to manually see the server’s SIM card IP address and update it in the client’s source code or configuration file every time the modem restarted, since the IP address changes with every network registration. With the online interface

each modem automatically stores the SIM card's IP address on every start up. The client while registering also checks if there are any predefined connections assigned to that modem. If there's any then the client will check which modem it is supposed to connect with, which SIM card that modem has and the SIM card's IP address. By doing all of this the client automatically loads the server's IP address even if it keeps changing, since all of the parameters are updated automatically. To connect two modems the user just needs to access the interface and add which modems are connected and everything will be configured automatically. The predefined connections' listing is pictured in figure 4.19. To insert a connection the user can access the interface pictured in figure 4.20.

Nova Ligação

Cliente	Servidor	Timestamp
CC Alexandre Herculano	SE Morgade	2013-03-25 16:21:58
354745033158462	SE Morgade	2013-03-27 18:33:37

1/1 10

Figure 4.19: Predefined connections listing

Cliente	Servidor
354745033158462	355632000891672
Versão do Software: SIEMENS TC65 REVISION 03	Versão do Software: SIEMENS TC65 REVISION 03
Baudrate: 115200	Baudrate: 115200
Timeout de Espera: 240000	Timeout de Espera: 240000
Frame Offset: 5	Frame Offset: 5
Inserir Ligação	


Figure 4.20: Insert predefined connection interface

The user only needs to choose one of the existing modems in the database to be the client and another to be the server. The other parameters are filled in automatically when changing the value of the drop down lists.




4.4.2.5 Event Visualization

The online interface also features an event visualization tool. This allows the user to see all the events of every modem registered in the database. There are five types of events: start, restart,

error, reception and transmission. The restart event is issued when the modem is about to shut down and restart the application. The start even occurs when the modem registers itself on the database. The error event signals that an unexpected error has occurred and it is normally followed by a restart event. The reception and transmission events indicate how many frames were sent and received since the last report. Through this interface the user is able to see the network's activity and easily monitor any modem or connection. The event table is pictured in figure 4.21.

 Sincronizar automaticamente

Modem	Ligação	Dispositivo	Tipo	Informação	Timestamp
355632000891672	14	Cliente	Reboot	Reset do modem	2013-04-09 16:57:10
355632000891672	14	Cliente	Rx	Receção de 0 tramas	2013-04-09 16:57:08
355632000891672	14	Cliente	Tx	Envio de 0 tramas	2013-04-09 16:56:54
354745033158686	14	Servidor	Reboot	Reset do modem	2013-04-09 16:53:03
354745033158686	14	Servidor	Rx	Receção de 0 tramas	2013-04-09 16:53:00
354745033158686	14	Servidor	Tx	Envio de 0 tramas	2013-04-09 16:52:55
355632000891672	14	Cliente	Rx	Receção de 3 tramas	2013-04-09 16:52:36
355632000891672	14	Cliente	Tx	Envio de 0 tramas	2013-04-09 16:52:32
354745033158686	14	Servidor	Rx	Receção de 0 tramas	2013-04-09 16:50:41
354745033158686	14	Servidor	Tx	Envio de 3 tramas	2013-04-09 16:50:41

 1/35  10 

Legenda: Start Restart Erro Receção Envio

Figure 4.21: Events visualization

The user can order the table's entries by each column and navigate through the results using the navigator below the table. It also features automatic synchronization. When activated the table is automatically updated with new events. The events can also be filtered by modem, connection, device type, event type and date. The filter interface is pictured in image 4.22.

4.5 Performance Considerations

In this chapter it will be detailed the performance considerations implemented while designing the Java application. The main resources optimized were: CPU usage, system's memory and communications delay. This application runs on a wireless module with limited resources. The capabilities available on this kind of devices are considerable more restricted than a regular PC. The modem has a very limited storage space for Java applications (approximately 1,5MB), a restricted memory and a rather limited processing potential [31]. The modem's processor may stress with significantly complex or poorly designed applications introducing excessive delay or even fail to run due to memory shortage. On the other hand, considering the whole system's architecture, the Frontend's timeout is a critical factor. If the Substation's response takes longer than the Frontend's

Filtragem

Modem: 355632000891672

Ligação: 14

Dispositivo: Todos

Sincronizar automaticamente

Tipo: Todos

Data: 2013-04-03 até 2013

Abri, 2013

Modem	Ligação	Dispositivo	Tipo	Informação	Timestamp
355632000891672	14	Cliente	Reboot	Reset do modem	2013-04-08 15:02:48
355632000891672	14	Cliente	Rx	Receção de 0 tramas	2013-04-08 15:02:38
355632000891672	14	Cliente	Tx	Envio de 0 tramas	2013-04-08 15:02:38
355632000891672	14	Cliente	Rx	Receção de 0 tramas	2013-04-08 14:59:40

Figure 4.22: Event filtering

timeout to arrive, the Frontend will discard the response and resend the frame, making the communication unviable. The round-trip time has to be lower than the Frontend's timeout, otherwise the frames will be discarded. The company's Frontend's parameters are very demanding [28] and although these parameters can be changed, there was a concern to optimize the application's performance in order to fulfill these requirements as best as possible.

4.5.1 J2ME and Jar Size

The modem uses the J2ME environment. It was originally designed to manage the constraints related with application development in small devices. It features the basics for Java ME technology and it's suitable to run on devices with limited power capacity, memory and display [27]. Although this is the most suitable environment for this kind of modems, it has a rather limited range of available libraries [27]. This made some objectives hard to accomplish. For example the functionality to write and read text files from the modem's file system would be trivial to achieve using the standard Java API, through for example `FileReader`. However this class isn't available in J2ME, neither the most common methods for file manipulation. One possible solution explored would be adding the Java classes' full specification to the application's project. This however increased the executable size by approximately 200KB. Since the initial executable size, without the additional specification, it's around 30KB (2% of the total available space) this would increase its size by 670%. Bearing in mind the full storage capacity is approximately 1,5MB it would mean that the executable size would occupy 15% of the total available space instead of the initial 2%. The comparison between the basic J2ME specification and the solution that involves adding additional Java libraries (represented by J2ME+) is pictured in figure 4.23.

Although adding additional libraries has obvious advantages like adding additional functionalities to the application and simplifying the coding process, increasing the file system's occupied

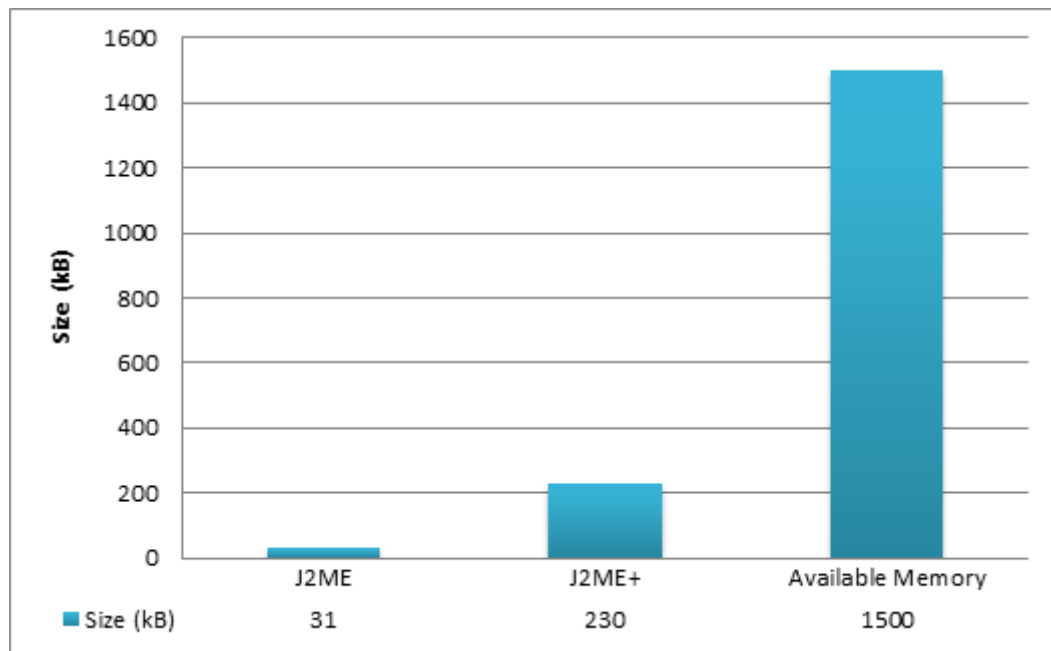


Figure 4.23: Solution's occupied space comparison

space by 13% and the executable size by 670% would be completely unacceptable. Instead ways were sought to do the same functionalities without adding an excessively amount of unused space, thus optimizing the storage capacity. Using the connector interface available in J2ME (the same interface used to open socket connections), it is possible to read and write text files although in a more intricate way. This however manages to minimize the executable size by maintaining the regular J2ME classes. Besides this and bearing in mind the modem's memory constraints mentioned before, the executable size was also minimized through obfuscation. An obfuscator is a program that removes all dispensable information from the compiled Java program [32]. Usually it is used as a security method to make the application more difficult to reverse engineer. However it is still useful for minimizing the executable size, removing all the unnecessary information makes the executable file smaller. The size reduction depends on the obfuscator and the application, but at least a 10% reduction is expectable [33]. In this case it reduced the file size from 31.243 to 21.722 bytes, reducing it by roughly 30%. Besides obfuscation the size was also reduced by using an unnamed package. The references to the classes in Java are done using their fully qualified name [32], which can add a significant overhead when using long class names. Using a package without a name eliminates this overhead. Although this is a detail and the impact on the file size is very small, it decreased the executable size from 21.722 to 21.456 (about 1,2%). The space reduction is presented in figure 4.24.

By minimizing the final file size the modem's resources are optimized and can be used for other useful purposes like store instruction files, other applications or different versions of the same application.

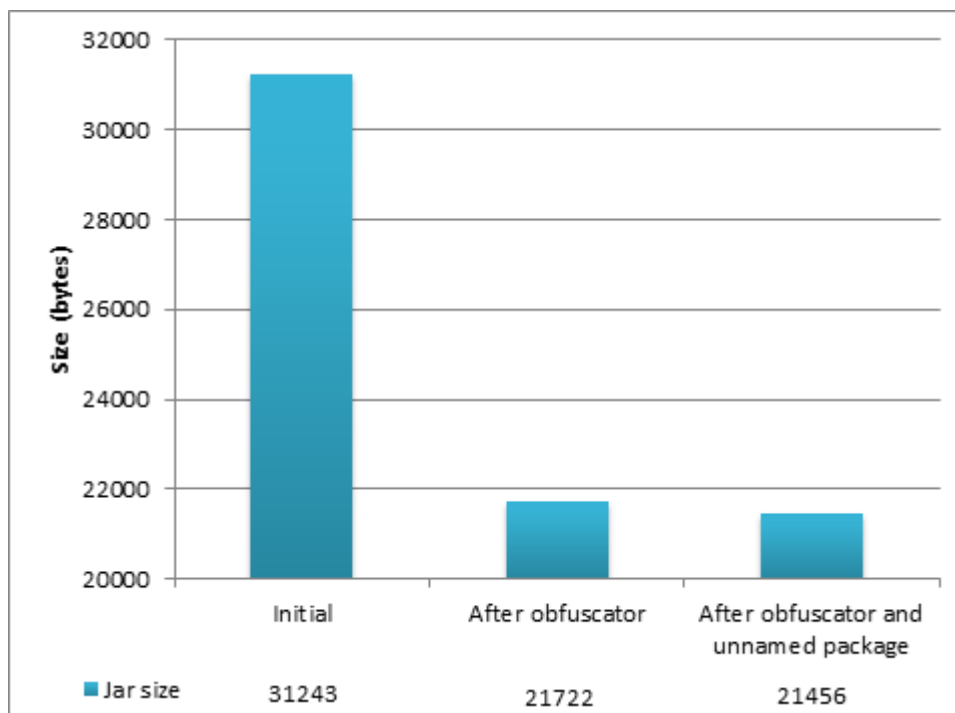


Figure 4.24: Occupied space after size reduction techniques

4.5.2 Busy Loops

Generally when the application needs to actively wait for something (like a busy loop) it blocks the execution thread. As a good practice, this should be done in a separate thread to avoid hanging the main thread [33]. However this practice wasn't adopted in this application. Throughout the application there is one point where the application blocks its main thread and two busy loops are used in a polling fashion in different threads: in the server's application while waiting for a socket connection and in both applications the reception and transmission threads poll with a busy loop the mobile network and the serial port respectively. To create a socket connection the server uses the `ServerSocketConnection` interface. This interface uses the `acceptAndOpen` method which blocks the execution thread until an incoming connection is detected [27]. This method is called in the initiation phase in the main thread. As a good practice this method should be executed in a different thread to avoid hanging the main application. In this particular application this is unnecessary. This would make sense if the application would run on a user interface because the user would actually see the application hanging and not responding, since the modem does everything internally and there is no user interface, there is no need. It is more adequate to spend the processing capabilities to check for a client connection than wasting it on two separate threads where one of them just kept the main thread alive. The server application also needs a client connection to continue its execution. It is important to remember that the application's main purpose is to establish a connection, so at that point there's nothing more important than detecting a client connection. The transmission and reception thread detect new information available in the TCP and

serial port connection with the input stream's available method. This method indicates how many bytes are available in that connection [27]. The application queries with a busy loop the amount of available data. This is rather inefficient and may lead to low performance. Instead applications should use the reading methods that specify the amount of data to be read in a more intelligent way [33]. Although this is a good practice in general applications driven by user interfaces it isn't in this particular case. Querying in a busy loop how much data is ready uses a high amount of the available processing time just checking the number of bytes that can be read while most of the times there's none. This might not be desirable in other applications but it is in this one. Since the main functionality of the application is to manage the flux of information it makes sense that most of the processing is destined to this purpose. In this case this is the main functionality, everything else is secondary. It is also important to minimize the delay the processing introduces in the whole connection, by constantly pooling the communications channels the data is detected faster and interpreted sooner thus reducing the processing delay. Other approaches might be more efficient for other functionalities in different applications (like animations, processing operations or graphical interfaces) but in this one this is the most adequate for this solution.

4.5.3 Garbage Collection

The garbage collector allows the automatic memory management enabling increased abstraction of interfaces and more reliable code. Its main responsibilities are memory allocation, guaranteeing that any referred objects remain accessible in the memory and freeing memory used by unreachable objects from references in executing code. The JVM initially requests a certain amount of space from the operating system. The garbage collector uses part of this space, called the heap, to allocate the program's objects [30]. Objects can be either referenced or no longer referenced. No longer referenced objects aren't needed and the resources allocated to them can be freed. The set of procedures that find and free the resources allocated to these objects is called garbage collection. Garbage collection involves several complex processes which take time and resources of their own and might introduce a significant delay [30]. Skillful programming is able to reuse existing objects instead of creating new ones, avoiding the accumulation of garbage objects on the heap. Doing this it is possible to minimize the time each garbage collection takes, making the application's execution faster. Reusing doesn't solve everything though. Immutable objects can't be changed after creation and easily become garbage objects if their initial value is not needed or changes. Also immutable objects are quite common and frequently used to guarantee thread safety and code reliability. Most programmers forget how frequently immutable objects are created and the number of garbage objects they produce. Each time an immutable object's value changes a new immutable object must be created with the new value [33]. The initial object is now considered a garbage object and must be garbage collected. Many times the advantages of immutable objects are not worth the cost and it is preferable to use reusable objects instead. One common example is the `java.lang.String`. The `String` object is immutable and many applications overlook the number of garbage objects it can generate even in simple operations. The developed applications were

tuned in order to minimize the garbage collection effort. One example from the client's application where the String object was optimized is pictured in figure 4.25. The code below was used in the client application to print a String with the characters read from the serial port connection. All the available bytes are read and stored in a ByteArrayOutputStream and then converted to a byteArray. This is a clear example of the implemented code of something that happens frequently.

```
1338   int len = dataByte.length;
1339   String stream = "";
1340   for (int i = 0; i < len; i++) {
1341       stream = stream + (dataByte[i] & 0xFF);
1342       stream = stream + " ";
1343   }
```

Figure 4.25: Source code before performance tuning

The dataByte variable stores the bytes read from the serial port. The assignments in line 1341 and 1342 don't actually modify the string stream, because String is an immutable object. Instead it creates a new String copying the contents assigned on line 1341 and it creates a new one again on line 1342, with an additional space. So with every iteration of the For loop, it is created two new String objects. In total it unnecessarily creates $len * 2$ garbage objects. For example if the number of bytes read from the serial port is 128, there will be 256 garbage objects that need collection. Problems related with the String object can be solved using the java.lang.StringBuffer. StringBuffer represents a mutable sequence of characters [27], being similar to the String object but it can be modified. It is possible to use this object instead of using the String object, avoiding the creation of the additional garbage objects. This example can be implemented much more efficiently using StringBuffer exemplified in figure 4.26.

```
1335   int len = dataByte.length;
1336   StringBuffer stream = new StringBuffer(len);
1337   for (int i = 0; i < len; i++) {
1338       stream.append(dataByte[i] & 0xFF);
1339       stream.append(" ");
1340   }
```

Figure 4.26: Source code after performance tuning

By implementing the StringBuffer object the same object is used on each assignment, by simply appending the new information to that object, therefore only one object is used instead of 256. It is still important to note that this is a very meticulous approach, most of the times immutable objects are worth the cost and creating a few garbage objects doesn't have a noticeable impact on performance [32]. Even a virtual machine designed for smaller devices can comfortably handle thousands of objects per second [33]. It would be still worth to ask exactly what the cost of

using immutable objects is or if in fact there is a difference between the two solutions. In this case this upgrade managed to reduce the time of that part of the code by 82% (from 208ms to 38ms on average) when using 128 bytes. The tests measuring the delay of each solution, the performance improvement and the statistical comparison between the two are presented in chapter 5.1.

4.5.4 Multi-threading

Besides being necessary for implementing the full applications' specifications, multi-threading also makes the application more efficient and leads to a better performance. Multi-threading was already necessary for managing the information flux between the serial port and mobile network, since it is required a simultaneous two way flux of information. It also has an impact on performance because it allows a thread to execute while another is waiting on some condition, for example an HTTP response [33]. Sometimes simple thread usage isn't enough to fulfill the full multi-threading potential. It is important to remember that Java threading is not guaranteed to be pre-emptive but may be cooperative [30]. When several threads are executing simultaneously and one of them is waiting for a condition in a busy loop the processor might wait longer than normal before it changes to another thread, thus being cooperative. Instead every thread should call `yield` or `wait` voluntarily allowing the processor to cycle freely between the threads. Image 4.27 pictures one of the implemented mechanisms.

```

1158 while (TCPSocket_die == false)
1159     try {
1160         if (is_RS232.available() > 0) {
1161             synchronized(this) {
1162                 wait(50);
1163             }

```

Figure 4.27: Thread wait method implementation

This implementation optimizes the time spent on each thread, guaranteeing that none of the threads monopolizes the processor.

4.5.5 IEC 60870 Frame Validation

The transmission thread receives bytes from the serial port and processes them. This thread runs infinitely unless an error condition occurs. When there are new bytes available they are read and compared with the IEC 60870 standard's frame structure. Initially this comparison was implemented similarly to other serial port applications developed by the company under different protocols. In every iteration the available bytes at that time were buffered in an array and then the contents of the array compared with the IEC 60870 frame structure. However this solution works on the assumption that every byte of the frame arrives instantly. Although according to common experience the industrial PCs actually work like this, it's conceivable that they don't. In general

serial port applications even if all the bytes are sent as soon as possible they can be buffered by the operating system or the serial interface. In the application it was verified that sometimes the serial port's available method is called before the entire frame arrives. Since only part of the frame is compared with the frame structure, it will forcefully not have the standard structure, being considered as garbage and discarded. Every time this situation occurs the application would discard a correct frame. This situation is pictured in image 4.28. In situation 1 the industrial PC sends the whole frame. If no errors occurred in the serial port connection the frame has the correct IEC 60870 structure, being correctly validated by the Java application and then transmitted to the mobile network. The problem with the initial implementation is represented in situation 2. Unlike situation 1, the frame is not sent instantly. The Java application reads the frame's initial part, and compares the received data to the standard's structure. Since the whole frame wasn't available at that time it hasn't the correct structure because the trailer was never received, being the initial part of the frame discarded. The same happens when the rest of the frame arrives, since it has no header it is discarded also. Every time this situation occurred an otherwise correct frame would be wrongly discarded.

Instead using this approach it was implemented a new one that corrected this problem. It was used a state machine which interprets the frame byte by byte. Besides correcting this error state machines are more efficient [34]. The transmission thread reads a new byte in each iteration and it is immediately compared with the standard's structure. As the frame is received the state machine advances through the states, being each state a validation of the frame's structure. When a correct header is not received the byte is discarded being interpreted as garbage. A correct header makes the state machine advance through the states and when a trailer is finally received the frame is sent. States 1 to 5 validate variable length frames while states 6 and 7 validate fixed length frames. This solution is more efficient than the last and doesn't have the problem detailed before since the interpretation is made byte by byte instead on regular time intervals. The state machine diagram is pictured in figure 4.29. The `byt` variable represents the byte read on the serial port on that cycle. `L1` and `L2` represent the length of the frame specified in the standard. The standard's structure and its validation are detailed on chapter 2.2.4.

4.5.6 TCP vs. UDP

Network's speed can be evaluated in terms of bandwidth and latency [35]. Being bandwidth the rate of data transferred in an open connection and latency the time a single item of data takes to cross the network from the source to the destination. When there's a large amount of information being transmitted bandwidth is usually the most important factor, on the other hand if there's a need to transmit small amount of data latency usually has more effect on the networking speed [35]. In this application the deciding factor is latency since most of the times the information is exchanged in small packets. Also the time the frames take to reach the destination is crucial for the viability of the link due to the Frontend's timeout. It is also important to remember that both bandwidth and latency have an average and a variation, if the variation is large the latency may reach unacceptable values. J2ME supports two kinds of point to point connections, either UDP or TCP

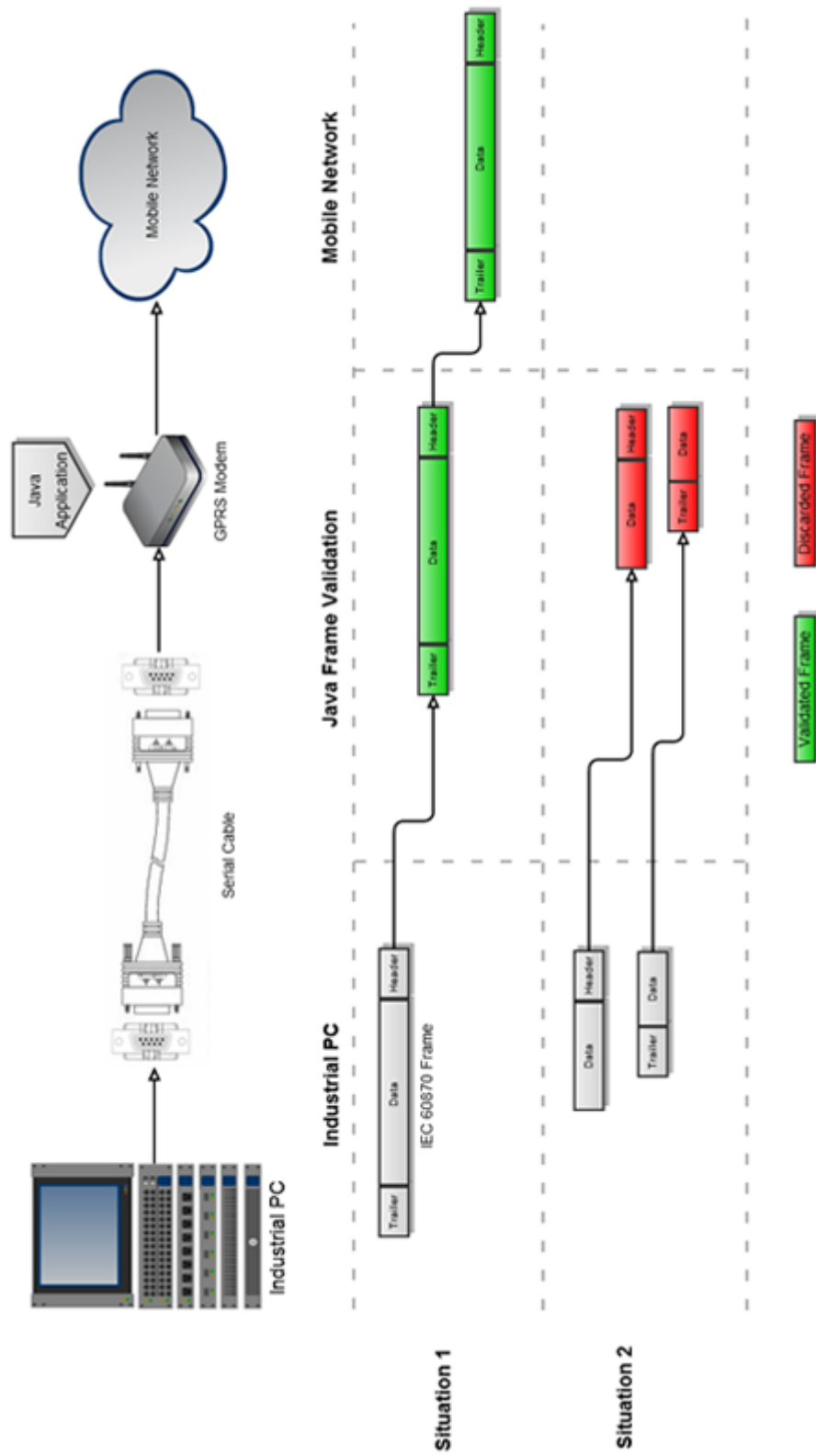


Figure 4.28: Initial frame validation sequence

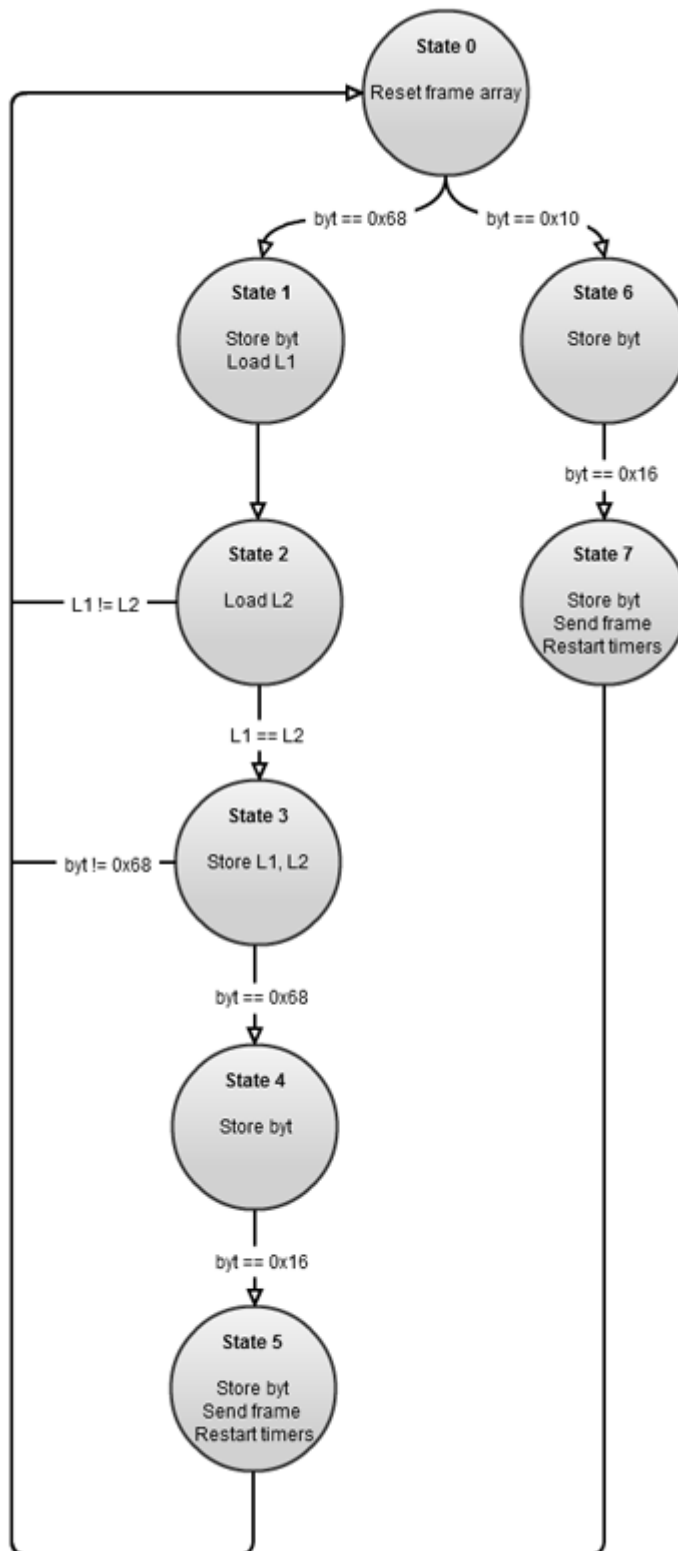


Figure 4.29: Frame validation's state machine

[27]. HTTP is also available but it's not suitable for machine to machine communications since an intermediate online server would have to manage the connection making the delay unbearable (HTTP latency is considerably higher than the latency verified in an UDP or TCP connection), being much more efficient one of the last two. Also it is not suitable to manage the flux of data online, outside the company's private APN due to security reasons. Having said this HTTP was still implemented to support the online monitoring interface but this interface is extra and doesn't have the same latency restrictions as the main connection. UDP connections are supported with the `UDPDatagramConnection` interface. The primary objective of this interface is to efficiently send data to the remote device without reliability, data order or a mechanism to prevent data duplication. There's no concept of request and response or connection establishment, the information is simply sent from the sender to the receiver. It also doesn't guarantee that the information reaches the destination but can send data faster than a TCP connection, in theory. A TCP connection is similar to a client-server connection, which is conceptually more suitable to this project since there's actually a master and a slave due to the IEC 60870 standard's configuration. The connection is implemented using the `ServerSocketConnection` and the `SocketConnection` interfaces. The first defines the server socket stream connection and the second the client socket stream connection. These interfaces are connection-oriented, for the connection to begin the server has to wait and listen for client connections. TCP is reliable, guarantees data order, delivery and has inbound mechanisms to prevent data duplication, although being slower than UDP, in theory. Table 4.1 summarizes the characteristics available in J2ME of the two connections.

Table 4.1: J2ME TCP/UDP comparison

Characteristics	TCP socket connection	UDP datagram connection
Speed	Lower (in theory)	Higher (in theory)
Data Ordering	Yes	No
Reliability	Yes	No
Duplication Prevention	Yes	No
Connection	Yes	No

The first version of the application used UDP. UDP was chosen because it was simpler to implement and minimized the delay. Although it fulfilled the requirements, TCP was conceptually more consistent and proved itself better than UDP. The tests measuring the delay of each connection, the performance improvement and the statistical comparison between the two are presented in chapter 5.2. Surprisingly and despite expectations, TCP revealed itself faster than UDP. This happens due to a number of reasons. TCP actually tries to buffer the data and fill a full network segment using the available bandwidth more efficiently [35]. This has a large impact in the performance when the packets being transmitted are considerably small, like the ones used in this application. TCP implements the Nagle's algorithm reducing the number of packets that are sent to the network, improving TCP/IP's efficiency. The Nagle's algorithm combines several small frames and sends them all at once avoiding the network's congestion and minimizing overhead [35]. On the other hand UDP sends the packets immediately, congesting the network with nu-

merous small packets. This was verified when there were a lot of packets being sent in a small amount of time, the delay kept increasing until the packets were eventually dropped. The excessive delay on UDP connections is also explained due to the specific implementation of the `UDPData-gramConnection`'s available method on the modems. Every time this method is used it blocks for around 500ms after UDP datagrams have been received. These reasons make TCP around 50% faster than UDP on average making it a better choice than UDP. The test results demonstrating this difference are presented in chapter 5.2

4.5.7 HTTP Requests Minimization

HTTP requests typically take around 5 to 8 seconds in the first round-trip and 2 to 4 seconds in the following round trips [33]. This delay is considerably higher than a socket connection and the corresponding HTTP thread might add a considerable delay to the application. Since HTTP is used to implement the extra online monitoring and configuring interface it shouldn't excessively disturb the main functionalities or add an excessive delay. Considering this the number of requests to the online server was minimized. The application's initiation HTTP requests sequence is pictured in figure 4.30.

Initially the application started by sending the modem's and SIM's parameters to the online server, the first request represented in the previous picture. The second request was issued to allow the modem to obtain the configurations specified in the online server. The third was sent after the connection initialization and updated the current connection in the online database. After the performance optimization the new HTTP requests sequence is pictured in figure 4.31.

The first and second requests were joined together. Every time a modem registers its parameters it also automatically obtains the corresponding configurations for that modem, eliminating one request. This burdens the first request with additional processing but the time the server takes to process that request is much lower than the time it takes to make a new request, thus improving the performance. This was also applied in the regular reporting functionality; whenever information is reported the modem also receives the commands issued from the online interface. The time interval each modem reports its information can also be configured remotely.

4.5.8 Web Integration Protocol Efficiency

A web service was used to implement the online interface. To manage the communications between the server and the devices one could use either REST or SOAP. REST uses every unique URL to represent a single object being the contents of that object accessible through an HTTP request. A GET request can be used to access the object while a POST to modify it [36]. SOAP on the other hand is an industry standard, it features a clear protocol and has defined rules for each application. It relies on XML to define the overall structure of the message as well as the responses. The messages are sent through HTTP or HTTPS requiring a remote procedure call to be processed [36]. REST is considerably simpler and more flexible than SOAP, since the services are accessible to any device with HTTP support. SOAP requires a new XML specification and most

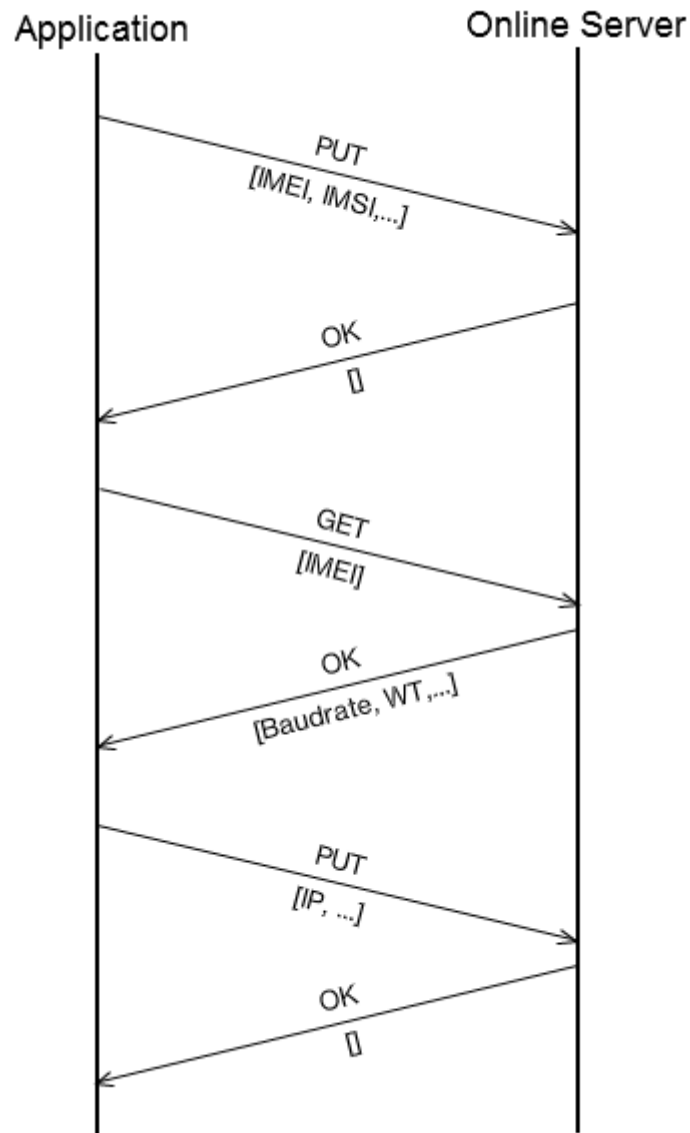


Figure 4.30: Initiation HTTP requests before optimization

of the times a SOAP toolkit is needed to parse and form requests. Considering security, while is always possible to distinguish the intent of each message by analyzing the HTTP request when using REST, in SOAP it's not possible to know the purpose of each message without consuming resources especially for that task [36]. On the other hand sensible information should never be transmitted through the URL like it is in REST. The interface between the modem's Java application and the online database was implemented using a REST interface through a custom protocol. The advantage of using REST instead of complex XML based protocols like SOAP is the reduced parsing time and the reduced overhead [36]. These characteristics make REST the most suitable protocol since the delay to process the request is crucial for the application's performance and the reduced overhead reduces bandwidth costs. Instead of using XML or other similar complex

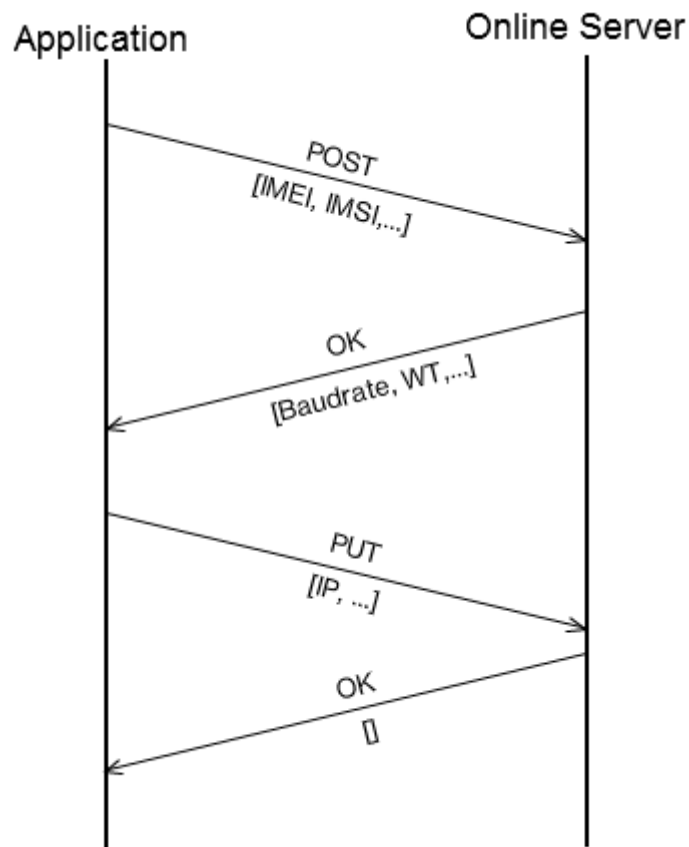


Figure 4.31: Initiation HTTP requests after optimization

protocols, a custom protocol was designed in order to improve the efficiency. An example of a possible XML object structure for this application is pictured in figure 4.32.



Figure 4.32: Example of an application's XML object

XML introduces an excessive unnecessary overhead unless the application actually needs the advanced capabilities of XML, which in this case it doesn't. It is possible to greatly reduce this overhead by designing a custom protocol and combining it with REST. The implemented protocol is pictured in figure 4.33.

```
20 | IMEI=355632000891672&Info=Siemens+TC65&Baudrate=115200&WT=240000  
21 | &FO=50&IMSI=268032200092972&Signal=30.99&IP=93.102.203.165:1025
```

Figure 4.33: Example of the implemented custom protocol

By designing both the protocol and the REST interface it is possible to parse the custom protocol and retrieve the required data. The custom protocol is able to greatly reduce the overhead, minimizing the delay while maintaining the required functionalities.

4.5.9 Automatic Modem's Geographic Location Detection

The online interface automatically displays the modem's geographic location on the map. If this location is changed the interface is automatically updated also. The modems have no GPS functionality and it is impossible to use a straightforward way to find the modem's location. To implement this functionality there was four possible alternatives: install a GPS device, obtain the location based on the SIM card's IP address, using the mobile network provider's cell IDs or through the local address transmitted in the IEC 60870 standard.

4.5.9.1 Install a GPS device

It is possible to simply connect a GPS device to the modem and obtain the modem's GPS location by accessing to that device. Once the information is available it can be sent to the online database through one of the HTTP requests. Also with this alternative it is possible to access the modem's location at any time. However this has a major disadvantage; it is necessary to buy the GPS device increasing the costs of the solution. Due to this reason the other solutions are more attractive, never the less it could become more relevant if the solution was applied to a mobile system, for example a mobile Substation.

4.5.9.2 SIM card's IP address

A single IP address is assigned every time a computer connects with the Internet. This address can be either dynamical or static, either way the assigned address can be traced to a specific geographic location. Unless a proxy is used, the obtained location is reasonably accurate [15]. However when using a GPRS connection the situation changes. The GGSN is responsible for assigning a GPRS IP address to a particular mobile unit. The IP addresses assigned by GGSN belong to the service provider's GPRS network, so even if the mobile device accesses the GPRS network while roaming or outside the area assigned to that provider, the address will still give the impression that the user

is still within the provider's network [15]. Due to this reason the GPRS IP address is not an accurate way to find the modem's geographic location.

4.5.9.3 Provider's cell ID

The signal coverage in the radio mobile network is guaranteed with a number of towers dispersed throughout the map. The network is made up of adjacent cells centered in each tower. Each cell is uniquely identified by four parameters: the cell ID, the code of the area that cell belongs to (area code), the code of the national network (MCC) and the company code which identifies the provider's company (MNC) [15]. This situation is pictured in figure 4.34.

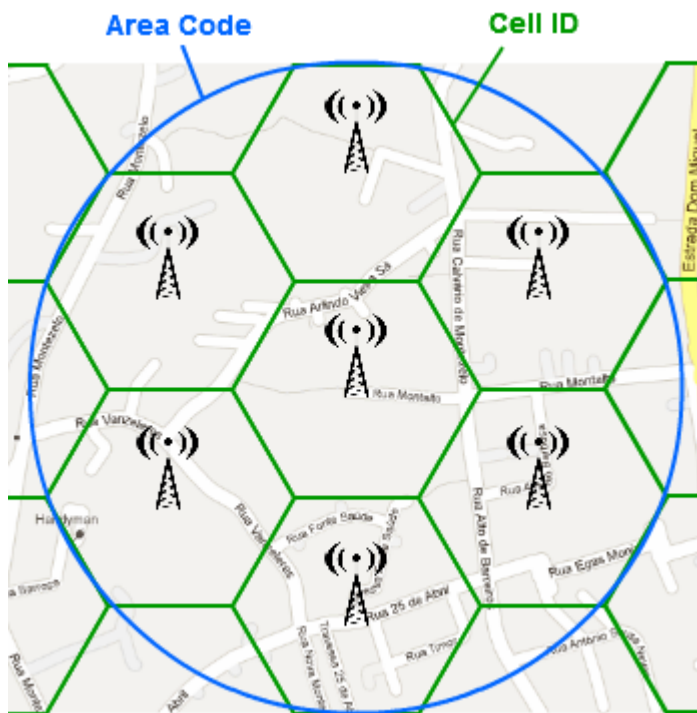


Figure 4.34: Mobile network geographic structure

It is possible to obtain all of these parameters using the modem's AT command interface and use them to obtain the mobile device geographic location. When the modem detects several towers the location can be reasonably accurate by triangulating the position. This method has a major disadvantage though, the cell ID's location is highly proprietary. There are some online databases with the cells' locations and even services capable of knowing the maximum distance allowed between a cell and a device before the device connects to a new cell, capable of detecting the location with accuracy down to 50 meters but they are very incomplete and many times outdated. These databases are user made and operators can easily protect their property by switching the identifiers at random. Although this would be a very interesting method with the right resources, the lack of a consistent database with the cells' location make it unviable.

4.5.9.4 IEC 60870 Local Address

According to the IEC 60870 standard specification every device has a local address and this address is sent in every frame [1]. Unfortunately this address isn't unique for every device inside the company's network. But the combination of the Substation's address with the Frontend's address is. Since the application receives and verifies the standard's frames it is possible to obtain both addresses. By comparing each pair of addresses with the company's database it is possible to know exactly to which Substation and Frontend each modem is connected. Once this information is available it is possible to know the modem's exact geographic location since the company has the location of every Substation and Frontend. This was the implemented solution to automatically obtain the modem's geographic location. When one frame from the Substation and one from the Frontend are transmitted in the same connection, both addresses are registered and compared with the installations database. By knowing to which specific Substation and Frontend each modem is connected it is possible to know their location since it is the same as the Substation's and Frontend's location. This solution still has some limitations since it is impossible to know the modem's location when it's not connected to the installation. Also if the Substation's industrial PC isn't in the correct place for example under maintenance, the obtained location is incorrect. Still under normal circumstances it is the best way without additional costs, to accurately find the geographic location.

4.5.9.5 Overview

Table 4.2 compares all the methods detailed before. It compares how much it costs to implement that method, how accurate is the obtained location and conceptually how often the location is available.

Table 4.2: Geographic location detection method comparison

Method	Cost	Accuracy	Availability
GPS Device	High	High	High
IP address	None	Very low	High
Cell ID	None	Medium	Low
Local Address	None	High	Medium

The implemented method, the IEC 60870 local address detection is the costless solution with highest accuracy, although with not the best availability due to the limitations mentioned before. Never the less in normal situations this is the best way to obtain the geographic location.

4.6 Satellite Systems Benchmarking

This section explores the best satellite systems and their services for implementing Power Substations telecontrol. It is presented a cost comparison between the main services available. This section wasn't overly detailed due to the lack of interest of the company in keeping exploring these

solutions due to the elevated associated costs. Instead the focus for this dissertation was to develop the system using GPRS modems. Never the less a brief overview is presented since it is in the initial dissertation's objectives. It is important to note that the information about the satellite systems delay, prices and even their services was exceptionally hard to find. Most providers require the customers to contact them directly to obtain this information and won't provide it unless there's an actual interest in buying a service. Also the prices might change according to the customer and the type and size of the application.

4.6.1 Iridium

Iridium uses 66 low earth orbiting satellites and has a pole to pole coverage. It supports a wide range of services like full duplex, real time calls for voice and data, SMS and Short Burst Data [24]. Its terminals can make and receive dial-up calls similarly to any landline modem with a 2400 bits/s rate. The setup time can take up to 40 seconds and the costs around €0.8 per minute. The latency and cost can be lower when both terminals are using the Iridium system. If the device is using the TCP/IP stack it is possible to make a point to point protocol dial-up connection to the internet. This has a lower throughput due to the extra overhead introduced by TCP/IP but considerably reduces the setup time. There's also the RUDICS service which is similar to a dial-up connection but it uses an internet connection rather than a dial-up modem. The setup time is greatly reduced and it is cheaper than dial-up having a cost of €0.5 per minute but it has a one-time fee of €1919. It can reduce costs in applications where it is required to monitor multiple devices [24, 37]. Besides the call based services mentioned before Iridium also supports message based services. The short burst data service is aimed at terminals that make frequent short connections transmitting messages up to 1960 bytes. It has a cost of €10 per month, €0.03 for 30 bytes and €0.0012 for the subsequent bytes. There's also the option to pay a higher €12 monthly fee and get the first 12000 bytes included. Besides the short burst data Iridium also supports the traditional GSM based text messaging carrying 160 characters with cost of €0.34 per message. Finally Iridium also supports the OpenPort service which offers a data service at 32, 64 and 128 kbits/s with a much lower cost than the regular price per byte than the traditional services. The OpenPort data costs vary depending on the monthly subscription but normally the price is between €4 and €13 per megabyte. The service is only supported on a specific terminal with a cost of €3070 [24, 37].

4.6.2 Inmarsat

Inmarsat has a fleet of geostationary satellites that cover most of the earth except the poles. It has a range of products and services aimed at both sea and land based users. It provides the BGAN, FleetBroadband, Fleet 33/55/77 and IsatM2M [25]. BGAN is a portable broadband product, which supports voice calls and IP data at speeds up to 492 Kbit/s. BGAN's line rental costs around €38 per month and has a fee of €5 per megabyte. The terminals cost between €1765 and €3454 depending on the specification. While BGAN is more adequate for land applications,

Inmarsat also has a marine based product called the FleetBroadband. It is physically larger, uses a stabilized antenna and has a top bitrate of 432Kbit/s. Unlike BGAN the line rental has no cost but charges €9 per megabyte subject to a minimum monthly spend of €23. The Fleet services offer voice, fax and dial-up data services. It uses MPDS a packet based, pay by the bit data service. Fleet 33 operates at 28Kbit/s uplink and 64Kbit/s downlink, Fleet 55 at 64Kbit/s uplink and downlink and Fleet at 128Kbit uplink and downlink. MPDS data costs €26 per megabyte, dial-up data costs €5 per minute for 64Kbit/s and €10 per minute for 128Kbit/s. Fleet terminals cost around €5756 for Fleet 33, €9824 for Fleet 55 and €12740 for Fleet 77 [25, 37]. Inmarsat also offers the IsatM2M service which is a burst data service that can send 10 or 25 byte messages and receive up to 100 byte messages. Each message has a cost of €0.046 per 10 byte message and €0.09 per 25 byte message. Each terminal has a minimal monthly spend of €4 and cost roughly €690 [25, 37].

4.6.3 Orbcomm

Orbcomm has 29 low Earth satellites and provides a message based communication service. The messages are downlinked by ground stations and can be delivered close to real time unless the satellite isn't in range. When this happens the messages are stored and sent as soon as the next satellite comes in range [22]. It provides services with unrestricted traffic for a €46 monthly fee with 2.4Kbit/s uplink and 4.8Kbit/s downlink. Terminals cost between €154 and €307 depending on if they are just modems or if they include a programmable microcontroller to collect data from other devices [22, 37].

4.6.4 Thuraya

Thuraya operates several different sectors offers voice and data satellite communications solutions. The satellite system covers Europe, south-east Asia, Australia, most of Africa and the Middle East [23]. It offers a dial-up data service at 9.6Kbit/s, packet data similar to GPRS at 60Kbit/s downlink and 15Kbit/s uplink and SMS messaging. A dial-up connection has a €27 monthly subscription and a cost of €1.5 per minute. A packet data connection has a €42 monthly subscription including the first 5MB and then a cost of €4 per MB [23, 37]. It also has the ThurayaIP service which offers internet access at roughly 450Kbit/s having a monthly cost of €422 for 138MB or €3 per MB. There's also a plan with no cost per MB for €3852 per month. The terminal is portable, small, weights 1.3kg and costs €3072 [23, 37].

4.6.5 Globalstar

Globalstar uses a low Earth orbit satellite constellation that covers North America, South America, Europe, north Africa and most of the Atlantic [26]. It offers dial-up connection with bitrate at 9.6Kbit/s, with a monthly fee of €30 and a cost of €0.7 per minute. Globalstar also has a short burst data service that supports messages up to 144 bytes. This service can be priced in 9 bytes or 36 bytes increments and start at €23 per month for 100 messages of 9 bytes [26, 37].

4.6.6 Overview

The main services referred before and their airtime charges are resumed in table 4.3 and their monthly airtime cost in table 4.4.

Table 4.3: Main satellite services airtime charges

System	Data rate (kbit/s)	Monthly fee	Charged rate
Iridium dialup	2,4	€ 11	€ 0,80/min
Iridium RUDICS	2,4	€ 11	€ 0,50/min
Iridium OpenPort	32/64/128	€ 27 to € 862	€ 4 to € 13/MB
Fleet MPDS	28/64/128	€ 0	€ 26/MB
Fleet 33 dialup	9,6	€ 0	€ 2/min
Fleet 55/77 ISDN	64	€ 0	€ 4/min
Fleet 77 ISDN2	128	€ 0	€ 5/min
BGAN	492	€ 38	€ 5/MB
FleetBroadband	432	€ 0	€ 9/MB
Thuraya dialup	9,6	€ 27	€ 0,80/min
Thuraya GmPRS	15	€ 42	€ 4/MB
ThurayaIP	444	€ 423	€ 3/MB
Globalstar dialup	9,6	€ 31	€ 0,80/min

Table 4.4: Main satellite services monthly airtime cost for a given amount of data

System	1MB	10MB	100MB	1000MB
Iridium dialup	€ 55	€ 458	€ 4.479	€ 44.694
Iridium RUDICS	€ 39	€ 296	€ 2.861	€ 28.516
Iridium OpenPort	€ 40	€ 97 (32 kbit/s)	€ 539 (32 kBit/s) € 620 (64 kbit/s)	€ 4314 (32 kbit/s) € 4853 (64 kbit/s) € 5755 (128 kbit/s)
Fleet MPDS	€ 26	€ 262	€ 2.619	€ 26.194
Fleet 33 dialup	€ 33	€ 331	€ 3.313	€ 33.127
Fleet 55/77 ISDN	€ 12	€ 116	€ 1.156	€ 11.556
Fleet 77 ISDN2	€ 11	€ 100	€ 1.001	€ 10.015
BGAN	€ 44	€ 92	€ 578	€ 5.431
FleetBroadband	€ 23	€ 92	€ 925	€ 9.245
Thuraya dialup	€ 39	€ 143	€ 1.183	€ 11.583
Thuraya GmPRS	€ 42	€ 64	€ 445	€ 4.259
ThurayaIP	€ 424	€ 424	€ 424	€ 3.852
Globalstar dialup	€ 42	€ 146	€ 1.186	€ 11.587

All of the satellite systems' coverage areas referred before includes Portugal [24, 25, 22, 23, 26]. The cheapest service is different for the amount of data used. For 1MB is Fleet 77 with a cost of €11 per month, for 10MB is Thuraya GmPRS with a cost of €64, for 100MB and 1000MB is ThurayaIP with a cost of €424 and €4259 per month respectively. It is expectable that the regular traffic between a Substation and Frontend is between 10MB and 100MB, so for this application

the best service would be the ThurayaIP. This solution would have a cost of €424 per month plus the price for two terminals at €3072 each.

Chapter 5

Conclusions and Tests

This chapter describes the tests that were conducted to assess the performance of the system, more specifically to verify if improvements had indeed been achieved as a consequence of the optimization performed in the system as detailed in chapter 4.5. It presents the results obtained, including measurements of average delays, analysing such results to understand and justify the observed system behaviour. A statistical analysis is presented, which is able to prove that there is a statistically significant improvement in the system performance. Results of tests conducted in a real world situation, using a simulated Frontend and a real Industrial PC of the Morgade Power Substation, are also described. This chapter also draws the main conclusions, pointing possible routes for future work and highlighting the current system's limitations, where improvement is still desirable.

5.1 Garbage Collection

Referring to the garbage collection optimization stated on chapter 4.5.3, the tests measure the time before and after the code pictured in this chapter, using both the String object and the StringBuffer object. To evaluate and compare the two solutions it was used a paired t-test. The purpose of the statistical evaluation is to verify if there is in fact evidence that the techniques to minimize the garbage collecting are improving the performance. This test is very effective for detecting differences when the test subjects are the same and when the subjects are measured before and after some sort of treatment or change [38]. Since the subject in this experience is the Java application and the objective is to measure the performance before and after the performance improvements, this is the most suitable test. It is important to size the number of tests that are needed to correctly draw conclusions about the impact of the technique in the application's performance. Assigning the Cohen's d (the anticipated effect size), the desired statistical power level and the significance criterion it is possible to estimate the minimum sample size required to make the test statistically relevant. The significance criterion represents the probability of mistakenly rejecting the null hypothesis [38], in other words the risk of concluding that the improvements were effective while

they weren't. The statistical power level represents the probability of rejecting a false null hypothesis [38], or the probability of rejecting the conclusion that there were no significant improvements when there were. The effect size represents the impact of a phenomenon [38], if it is expected that the improvements will have a small, medium or large effect. To estimate the number of tests it was used the standard values for the significance criterion (0.05) and the statistical power level (0.8) [38]. The effect size is the most difficult part to evaluate due to the generally low level of consciousness of the magnitude of the phenomena being the evaluation rather subjective [38]. It is particularly hard to estimate in this case since the performance of the garbage collecting also varies from device to device [32]. Considering this and the history of possible gain of efficient garbage collection [32] it is expected an effect size between medium and large, so it was used 0.6 (small being 0.2, medium 0.5 and large 0.8). The used parameters are resumed in table 5.1.

Table 5.1: Garbage collection tests sample size estimation parameters

Parameter	Value
Cohen's d	0.6
Statistical power level	0.8
Significance criterion	0.05

Tests can be either one tailed or two tailed, one tailed tests are appropriate when a difference in one direction is expected while two tailed are appropriate when a difference in any direction is expected. For this test both will be evaluated. Considering these parameters the sample size results obtained are resumed in table 5.2.

Table 5.2: Garbage collection tests sample size estimation results

Results	Sample Size (units)
Minimum total (one-tailed hypothesis)	72
Minimum per group (one-tailed hypothesis)	36
Minimum total (two-tailed hypothesis)	90
Minimum per group (two-tailed hypothesis)	45

The tests measured the time just before and after the code pictured in chapter 4.5.3, using the String object and the StringBuffer object. It was used a byte array of 128 bytes, both tests were on the same modem, using the exact same application and with exactly the same conditions. Considering the sample size results it was done 50 tests per group. The results of each test can be found on appendix A. The average and the standard deviation of each set of tests can be found in table 5.3.

Comparing the averages of both methods one can notice a dramatic improvement. The average time to run the code is 82% faster with the StringBuffer object and there's a reduction of 171 ms on average. Another noticeable difference is the standard deviation, which is considerably higher on StringBuffer's test. It is important to notice that the time it takes to execute the instructions varies

Table 5.3: Garbage collection tests results

	String Object	StringBuffer Object
Average	208 ms	37 ms
Standard Deviation	38,29 ms	64,24 ms

and there are always background tasks that take time, since the String object's delay is larger these tasks are less visible thus making the standard deviation smaller. It is still necessary to prove that these tests weren't a fluke and are statistical relevant. To do so it was used a paired t-test for the mean. The results are resumed in table 5.8.

Table 5.4: Garbage collection paired t-test results

Parameter	Results
t Stat	14,32
P(T<=t) one-tail	1,96E-19
t Critical one-tail	1,68
P(T<=t) two-tail	3,91E-19
t Critical two-tail	2,01

From the results it is observable that the absolute value of the t Stat is greater than the t Critical one-tail and the t Critical two-tail. So it is safe to assume that there is a significant difference between the two tests (based on the two-tail) and the String's mean is larger than the StringBuffer's mean (based on the one-tail) [38]. It is also correct to say these tests weren't a coincidence, since the probability that these tests were a fluke, given by the P one-tail and two-tail is much smaller than specified significance criterion (0.05). Considering this, it is safe to reject the null hypothesis and conclude that there is an actual statistical difference between the String tests and the StringBuffer tests. And since the StringBuffer's average delay was definitely smaller than the String's average delay the t test also proves that the StringBuffer's delay is statistically significantly smaller, therefore better in a performance perspective. Remembering the initial statement on chapter 4.5.3 that a few garbage objects didn't have a noticeable impact on performance, well this is also true. Although there was a statistically significant improvement and on average the improvements are 82% better than the initial solution this means a difference of 171ms on average. It is discussable how critical this is on the whole system. Since the delay would be smaller for smaller frames which are the most common in the standard and such a high garbage collection cost isn't often present in the code, so the delay wouldn't reach such high values very often. Either way this is a costless improvement always worth to implement.

5.2 UDP and TCP

Referring to the garbage collection optimization stated on chapter 4.5.6, the tests measure the round-trip delay time. It was impossible to measure a single trip's delay from the sender to receiver since the modem's internal clock is desynchronized. Each modem's internal clock has a different time so it was impossible to accurately compare both of the modem's clocks with a precision of milliseconds. Instead it was measured the round-trip delay time, one modem recorded the time it sent a single frame, the remote modem simply received and sent back to the sender and the initial modem recorded the received time. Similarly to the previous chapter, to evaluate and compare the two solutions it was used a paired t-test. The test's scheme is represented in picture 5.1.

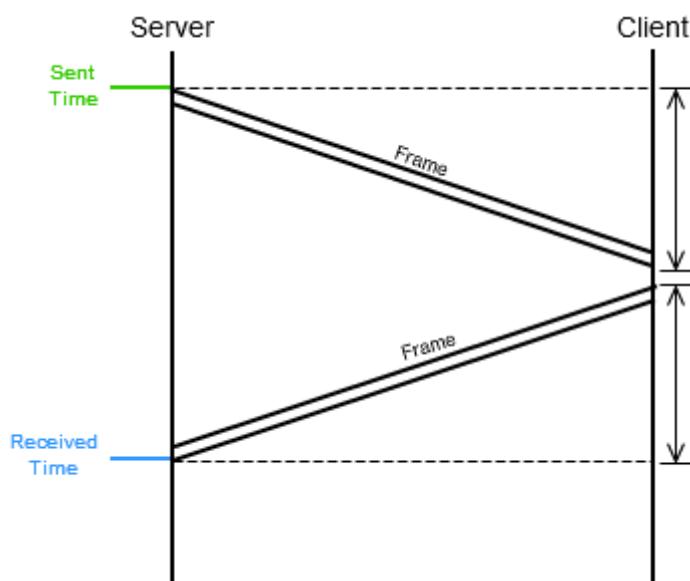


Figure 5.1: TCP/UDP tests scheme

To estimate the number of tests it was used the standard values for the significance criterion (0.05) and the statistical power level (0.8) [38]. Since it is expected an effect size between medium and large, closer to large, it was used 0.7. The used parameters are resumed in table 5.5.

Table 5.5: TCP/UDP tests sample size estimation parameters

Parameter	Value
Cohen's d	0.7
Statistical power level	0.8
Significance criterion	0.05

Considering these parameters the sample size results obtained are resumed in table 5.6.

It was realized 40 tests using the same modem and same application, with the exact same situations. They were done on two week days on the exact same time of the day (19:00), so

Table 5.6: TCP/UDP tests sample size estimation results

Results	Sample Size (units)
Minimum total (one-tailed hypothesis)	52
Minimum per group (one-tailed hypothesis)	26
Minimum total (two-tailed hypothesis)	68
Minimum per group (two-tailed hypothesis)	34

the network's congestion would be the same for both. The results of each test can be found on Appendix B. The average and the standard deviation of each set of test can be found in table 5.7.

Table 5.7: TCP/UDP tests results

	TCP Connection	UDP Connection
Average	2,013 s	4,198 s
Standard Deviation	0,274 s	1,502 s

Comparing the averages of both connections one can notice a significant difference; the UDP's average delay is considerably higher than TCP's. Changing the connection from UDP to TCP improves the delay on average by 52% and there's a reduction of 2,185 seconds. The TCP's standard deviation is also considerably smaller than UDP. This was expected since UDP doesn't guarantee reliability and the packets may take different paths each time [35] thus the high deviation. It is still necessary to prove that these tests weren't a fluke and are statistical relevant. To do so it was used a paired t-test for the mean. The results are resumed in the table below.

Table 5.8: TCP/UDP paired t-test results

Parameter	Results
t Stat	12,22
P(T<=t) one-tail	3,3E-15
t Critical one-tail	1,69
P(T<=t) two-tail	6,6E-15
t Critical two-tail	2,02

From the results it is observable that the absolute value of the t Stat is greater than the t Critical one-tail and the t Critical two-tail. So it is safe to assume that there is a significant difference between the two tests (based on the two-tail) and the UDP's average delay is larger than the TCP's average delay (based on the one-tail) [38]. It is also correct to say these tests weren't a coincidence, since the probability that these tests were a fluke, given by the P one-tail and two-tail is much smaller than specified significance criterion (0.05). Considering this, it is safe to reject the null hypothesis and conclude that there is an actual statistical difference between the UDP tests and the TCP tests. And since the TCP's average delay was definitely smaller than the UDP's average

delay the t test also proves that the TCP's delay is statistically significantly smaller, therefore better in a performance perspective. The advantages implemented by TCP plus the lower delay verified when comparing it with UDP makes TCP the best choice for this application. Implementing TCP significantly decreases the delay improving the whole application's performance by a substantial value.

5.3 Real Tests

These tests were done in a real situation using a Frontend and the industrial PC of the Morgade substation. It was used two modems Siemens TC65 using two SIM cards from TMN. It was necessary to do some minor adjustments in the serial port's configuration with the substation and the solution worked as expected. It was verified that the delay was higher initially and lowered when the connection stabilized. The delay when the connection was stable was around 2 seconds as expected. The Frontend's logs about this test can be found on Appendix C. The logs show the correct communication of the IEC 60870-5-101 standard of both Frontend's and Substation's responses and their timeline. Picture 5.2 presents the Frontend's configurations used for this test.

The main conclusion about the tests was that the Frontend's parameters might have to be adjusted according to the specific Substation where the solution is implemented. The Frontend's timeout has to be 5 or 6 seconds to guarantee the correct reception due to the delay's variation in the beginning of the connection. However if the timeout is configured to 3 seconds the connection will still be viable although the initial frames might be discarded until the connection stabilizes. Different substations might have different timeouts depending on how far they are. Many times the company's Frontends have timeouts of less than 1 second. It is impossible to accomplish a round-trip time this low using this solution, since it is impossible to achieve optic fiber's performance with a system of this nature. Never the less this solution is incomparably cheaper and faster to implement than an optic fiber based system and reveals itself a flexible and consistent alternative to the classic methods of telecontrol. Lastly, the tests demonstrated that the Frontend was able to correctly communicate with the Substation proving that this solution is a viable powerful alternative to telecontrol electrical substations.

5.4 Conclusions

The satellite solution wasn't implemented due to the lack of interest revealed by the company, never the less it also reveals as an interesting solution with moderate costs that should be considered more carefully in a near future. The system presents several advantages and interesting features when compared with the traditional communications technologies. It is a very flexible and adaptive solution that can be applied to different situations and many applications having no geographic limitations like optic fiber or microwave links as long as the operator's network covers that location. It is very portable since the modems are small and lightweight and have no cost to move the system from a specific location to another but the cost to move the modems, this can

IEC 870-5-101 Master Properties

Name:

Link Address:

Serial Port:

Retries:

Poll Interval: (milli seconds)

Read Timeout: (milli seconds)

GI Period: (seconds)

TimeSync: (seconds)

Test Function: (seconds)

Enable Balance Mode

Auto create tag from response messages

Protocol Field Size Parameters

Link Address: Sector:

IOA: COT:

Modem

Phone Number (ATD):

Dial Timeout: (seconds)

Enable Modem:

Figure 5.2: Frontend's real test configuration

be very troublesome when using optic fiber and microwave links. The system requires no previous configuration since it is plug-and-play and the link establishment is seamless. This system is considerably cheaper than implementing an optic fiber or microwave link. The online monitoring and configuring interface made the system even more accessible and endowed it with several new interesting features. The interface made the system accessible and configurable from virtually everywhere, provided a tool for the system's registration and supports the system's escalation. Also this interface revealed itself as a very interesting tool to be applied not just to the substation telecontrol but also to the telecontrol of the medium voltage network. This network currently has

3126 Siemens TC65 modems installed and there is no registration tool or a way to find out which modem is where or to see any information about each modem. The online monitoring interface implements a way to monitor, configure and manage all of these 3126 modems and their connections completely automatically. The interface was also built in a way that any device with an internet connection can use the services. This eases the scalability of the system and manages to make the system useful for future devices that may be included in the network. Overall the system has several interesting features that make it a very interesting alternative to telecontrol electrical substations and the online interface is a very powerful tool to be applied in the company's context. The additional features made the system even more interesting making it error resilient and adequate to be used in long periods of time. The performance improvements minimized the processing delay and optimized the occupied space making the only relevant delay the one introduced by the mobile network. Finally all the proposed objectives were accomplished and surpassed and the additional features and the online monitoring interface were introduced to further complete the solution. The developed system reveals itself as a viable and interesting solution to remotely control electrical substations. The real tests demonstrated that the communications channel can be supported using GPRS modems and this system represents a sustainable alternative to guarantee communications between a SCADA frontend and an electrical Substation.

5.5 Limitations

This system is limited to the operator's network coverage zone. If a particular substation is outside the operator's network it won't be able to support the connection. This however is high unlikely since the company has deals with two of the major mobile network providers and they provide coverage to most of the country. Also this solution is highly dependent from the provider's network. This makes the connection not fully manageable from the company's point of view and makes it dependent from an external entity. Besides this the system also has a minimal round-trip delay of two seconds which is the operator's network delay, the optic fiber delay is significantly lower.

5.6 Future Work

Despite all the new features introduced by the online interface the implemented functionalities could be further explored and implemented new ones since it has a great potential. The security considerations for the company's integration should be also carefully reviewed. A solution using a satellite system should also be further explored and tested as the benchmarking revealed that the costs weren't too excessive and might be an interesting option. A new project involving the conversion of IEC 60870 RS-232 data to IP also could be an interesting project since the future of SCADA communications involves internet communications [39, 6, 5, 9]. This could be implemented using a modem with an RS-232 port and an Ethernet port with a software to manage the

communications between the SCADA Frontend and Electrical Substations. This also falls with the company's strategy to convert the existing networks to IP.

Appendix A

Garbage Collection Tests

This chapter presents all the tests and their results to measure the delay introduced by garbage collection.

A.1 String Object Tests

Table A.1: String object measured delay

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
1	01:10:12	904	272
	01:10:13	176	
2	01:10:13	347	189
	01:10:13	536	
3	01:10:13	665	203
	01:10:13	868	
4	01:10:14	131	208
	01:10:14	339	
5	01:10:14	394	213
	01:10:14	607	
6	01:10:14	644	240
	01:10:14	884	
7	01:10:14	893	217
	01:10:15	110	
8	01:10:15	119	185
	01:10:15	304	
9	01:10:15	313	212
	01:10:15	525	
10	01:10:15	530	212

Continued on next page

Table A.1 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	01:10:15	742	
11	01:13:42	751	263
	01:13:43	14	
12	01:13:43	42	341
	01:13:43	383	
13	01:13:43	420	203
	01:13:43	623	
14	01:13:43	766	259
	01:13:44	25	
15	01:13:44	85	203
	01:13:44	288	
16	01:13:44	574	208
	01:13:44	782	
17	01:13:44	851	189
	01:13:45	40	
18	01:13:45	77	236
	01:13:45	313	
19	01:13:45	317	208
	01:13:45	525	
20	01:13:45	530	216
	01:13:45	746	
21	01:14:08	18	351
	01:14:08	369	
22	01:14:08	424	185
	01:14:08	609	
23	01:14:08	697	203
	01:14:08	900	
24	01:14:09	80	207
	01:14:09	287	
25	01:14:09	518	208
	01:14:09	726	
26	01:14:09	786	231
	01:14:10	17	
27	01:14:10	35	185
	01:14:10	220	
28	01:14:10	229	198

Continued on next page

Table A.1 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	01:14:10	427	
29	01:14:10	437	207
	01:14:10	644	
30	01:14:10	649	203
	01:14:10	852	
31	01:14:33	797	337
	01:14:34	134	
32	01:14:34	295	189
	01:14:34	484	
33	01:14:34	531	203
	01:14:34	734	
34	01:14:34	909	277
	01:14:35	186	
35	01:14:35	228	207
	01:14:35	435	
36	01:14:35	620	208
	01:14:35	828	
37	01:14:36	3	189
	01:14:36	192	
38	01:14:36	220	217
	01:14:36	437	
39	01:14:36	446	208
	01:14:36	654	
40	01:14:36	658	213
	01:14:36	871	
41	01:15:03	314	268
	01:15:03	582	
42	01:15:03	660	194
	01:15:03	854	
43	01:15:04	44	221
	01:15:04	265	
44	01:15:04	334	208
	01:15:04	542	
45	01:15:04	782	222
	01:15:05	4	
46	01:15:05	64	249

Continued on next page

Table A.1 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	01:15:05	313	
47	01:15:05	327	189
	01:15:05	516	
48	01:15:05	525	212
	01:15:05	737	
49	01:15:05	747	212
	01:15:05	959	
50	01:15:05	964	216
	01:15:06	180	

A.2 StringBuffer Object Tests

Table A.2: StringBuffer object measured delay

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
1	01:19:54	954	42
	01:19:54	996	
2	01:19:55	9	28
	01:19:55	37	
3	01:19:55	217	116
	01:19:55	333	
4	01:19:55	342	189
	01:19:55	531	
5	01:19:55	586	88
	01:19:55	674	
6	01:19:55	683	33
	01:19:55	716	
7	01:19:55	794	125
	01:19:55	919	
8	01:19:55	960	28
	01:19:55	988	
9	01:19:55	993	60
	01:19:56	53	
10	01:19:56	62	23
	01:19:56	85	
11	01:20:34	797	50
	01:20:34	847	

Continued on next page

Table A.2 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
12	01:20:35	207	88
	01:20:35	295	
13	01:20:35	332	32
	01:20:35	364	
14	01:20:35	489	28
	01:20:35	517	
15	01:20:35	521	37
	01:20:35	558	
16	01:20:35	613	24
	01:20:35	637	
17	01:20:35	646	120
	01:20:35	766	
18	01:20:35	881	102
	01:20:35	983	
19	01:20:35	992	23
	01:20:36	15	
20	01:20:36	24	23
	01:20:36	47	
21	01:20:56	876	101
	01:20:56	977	
22	01:20:57	0	185
	01:20:57	185	
23	01:20:57	217	37
	01:20:57	254	
24	01:20:57	374	33
	01:20:57	407	
25	01:20:57	416	304
	01:20:57	720	
26	01:20:57	840	60
	01:20:57	900	
27	01:20:57	965	125
	01:20:58	90	
28	01:20:58	99	23
	01:20:58	122	
29	01:20:58	131	32
	01:20:58	163	
30	01:20:58	173	23

Continued on next page

Table A.2 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	01:20:58	196	
31	01:21:23	406	41
	01:21:23	447	
32	01:21:23	461	32
	01:21:23	493	
33	01:21:23	595	37
	01:21:23	632	
34	01:21:23	641	37
	01:21:23	678	
35	01:21:23	844	65
	01:21:23	909	
36	01:21:24	140	32
	01:21:24	172	
37	01:21:24	218	134
	01:21:24	352	
38	01:21:24	416	24
	01:21:24	440	
39	01:21:24	449	60
	01:21:24	509	
40	01:21:24	527	28
	01:21:24	555	
41	01:21:45	949	115
	01:21:46	64	
42	01:21:46	83	32
	01:21:46	115	
43	01:21:46	129	198
	01:21:46	327	
44	01:21:46	406	32
	01:21:46	438	
45	01:21:46	553	47
	01:21:46	600	
46	01:21:46	613	254
	01:21:46	867	
47	01:21:46	904	134
	01:21:47	38	
48	01:21:47	70	28

Continued on next page

Table A.2 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	01:21:47	98	
49	01:21:47	103	27
	01:21:47	130	
50	01:21:47	140	27
	01:21:47	167	

Appendix B

TCP and UDP Tests

This chapter presents all the tests and their results to measure the delay introduced by using either TCP or UDP.

B.1 UDP Tests

Table B.1: UDP connection measured delay

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
1	00:57:55	700	6900
	00:58:02	600	
2	01:06:43	4	5543
	01:06:48	547	
3	01:07:26	532	7832
	01:07:34	364	
4	01:37:19	37	8104
	01:37:27	141	
5	01:38:04	147	7320
	01:38:11	467	
6	01:41:01	863	5704
	01:41:07	567	
7	02:02:05	149	8188
	02:02:13	337	
8	02:02:39	119	7864
	02:02:46	983	
9	02:03:18	899	5404
	02:03:24	303	
10	02:03:50	837	4062

Continued on next page

Table B.1 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	02:03:54	899	
11	02:04:14	459	3780
	02:04:18	239	
12	02:04:36	86	3753
	02:04:39	839	
13	02:04:56	888	3951
	02:05:00	839	
14	02:05:23	657	3762
	02:05:27	419	
15	02:05:51	631	4048
	02:05:55	679	
16	02:06:27	368	4611
	02:06:31	979	
17	02:06:51	128	4011
	02:06:55	139	
18	02:07:22	466	4293
	02:07:26	759	
19	02:07:54	968	4071
	02:07:59	39	
20	02:08:18	746	3573
	02:08:22	319	
21	02:14:49	100	7850
	02:14:56	950	
22	02:15:28	940	7970
	02:15:36	910	
23	02:15:55	723	5321
	02:16:01	44	
24	02:16:19	390	4740
	02:16:24	130	
25	02:16:48	389	4011
	02:16:52	400	
26	02:17:17	69	4071
	02:17:21	140	
27	02:17:38	69	3831
	02:17:41	900	
28	02:18:05	92	3923

Continued on next page

Table B.1 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	02:18:09	15	
29	02:18:30	749	4311
	02:18:35	60	
30	02:18:59	184	3716
	02:19:02	900	
31	02:19:30	509	4066
	02:19:34	575	
32	02:20:29	549	5271
	02:20:34	820	
33	02:20:51	989	4846
	02:20:56	835	
34	02:21:15	509	4371
	02:21:19	880	
35	02:21:38	572	3988
	02:21:42	560	
36	02:22:00	749	4186
	02:22:04	935	
37	02:22:23	129	4006
	02:22:27	135	
38	02:22:49	150	4210
	02:22:53	360	
39	02:23:25	169	4178
	02:23:29	347	
40	10:08:49	955	3923
	10:08:53	878	

B.2 TCP Tests

Table B.2: TCP connection measured delay

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
1	10:17:33	192	1690
	10:17:34	882	
2	10:20:06	312	1920
	10:20:08	232	
3	10:20:45	322	2335
	10:20:47	657	

Continued on next page

Table B.2 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
4	10:21:34	69	2188
	10:21:36	257	
5	10:22:34	189	2013
	20:22:36	202	
6	10:23:19	406	1796
	10:23:21	202	
7	10:24:04	392	1994
	10:24:06	386	
8	10:27:57	50	2170
	10:27:59	220	
9	10:28:31	924	1966
	10:28:33	890	
10	10:29:16	560	1984
	10:29:18	544	
11	10:29:58	929	1966
	10:30:00	895	
12	10:30:36	304	1911
	10:30:38	215	
13	10:31:17	529	1186
	10:31:18	715	
14	10:32:02	349	2326
	10:32:04	675	
15	10:32:46	121	2714
	10:32:48	835	
16	10:35:15	549	2026
	10:35:17	575	
17	10:36:06	203	2192
	10:36:08	395	
18	10:36:42	424	2211
	10:36:44	635	
19	10:37:21	253	1962
	10:37:23	215	
20	10:40:49	950	2585
	10:40:52	535	
21	10:41:22	170	1980
	10:41:24	150	
22	10:42:00	358	2312

Continued on next page

Table B.2 – Continued from previous page

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	10:42:02	670	
23	10:42:28	840	1984
	10:42:30	824	
24	10:43:02	573	1602
	10:43:04	175	
25	10:43:34	364	2345
	10:43:36	709	
26	10:45:24	884	1676
	10:45:26	560	
27	10:46:11	324	2211
	10:46:13	535	
28	10:46:35	897	1998
	10:46:37	895	
29	10:47:00	829	2151
	10:47:02	980	
30	10:47:29	827	2013
	10:47:31	840	
31	10:47:54	344	2031
	10:47:56	375	
32	10:48:48	829	2326
	10:48:51	155	
33	10:49:17	107	2188
	10:49:19	295	
34	10:49:43	784	1856
	10:49:45	640	
35	10:50:15	824	2331
	10:50:18	155	
36	10:52:01	570	1934
	10:52:03	504	
37	10:52:28	270	1740
	10:52:30	10	
38	10:52:53	960	1855
	10:52:55	815	
39	10:53:17	664	2216
	10:53:19	880	
40	10:53:44	124	2211

Continued on next page

Table B.2 – *Continued from previous page*

Test Number	Hour:Minutes:Seconds	Milliseconds	Delay (ms)
	10:53:46	335	

Appendix C

Real Tests Logs

This chapter presents the real test Frontend's logs. It is possible to see the message exchange, link initialization and the IEC 60870 message structure.

C.1 Log Sample

<- 17:45:44.671 [10 49 1E 67 16]

Status of link Request

Link Control: Status of link [PRM:1 FCV:0 FCB:0] LinkAdr: 30

-> 17:45:49.812 [10 0B 1E 29 16]

Status of link Response

Link Control: Status of link [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:45:49.828 [10 40 1E 5E 16]

Reset remote link Request

Link Control: Reset remote link [PRM:1 FCV:0 FCB:0] LinkAdr: 30

-> 17:45:55.265 [10 00 1E 1E 16]

ACK Response

Link Control: Acknowledge [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:45:59.281 [68 0A 0A 68 73 1E 64 01 06 01 00 00 00 14 11 16]

Interrogation Request

Link Control: User Data (Confirm) [PRM:1 FCV:1 FCB:1] LinkAdr: 30

ASDU: 100 <Interrogation Command> Count:1 SQ:0

COT: 6 <activation> Sector 1

QOI: 20 <Station Interrogation>

-> 17:46:04.906 [10 00 1E 1E 16]

ACK Response

Link Control: Acknowledge [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:04.921 [10 5B 1E 79 16]

Class 2 Request

Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:0] LinkAdr: 30

-> 17:46:06.171 [10 09 1E 27 16]

Data Not Available Response

Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:08.921 [10 7B 1E 99 16]

Class 2 Request

Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:1] LinkAdr: 30

-> 17:46:10.828 [10 09 1E 27 16]

Data Not Available Response

Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:12.921 [10 5B 1E 79 16]

Class 2 Request

Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:0] LinkAdr: 30

-> 17:46:15.281 [10 09 1E 27 16]

Data Not Available Response

Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

-> 17:46:16.921 [empty]
Request Timeout (No Activation Confirmation)

<- 17:46:20.921 [68 0A 0A 68 73 1E 64 01 06 02 00 00 00 14 12 16]
Interrogation Request
Link Control: User Data (Confirm) [PRM:1 FCV:1 FCB:1] LinkAdr: 30
ASDU: 100 <Interrogation Command> Count:1 SQ:0
COT: 6 <activation> Sector 2
QOI: 20 <Station Interrogation>

-> 17:46:23.703 [10 00 1E 1E 16]
ACK Response
Link Control: Acknowledge [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:24.921 [10 5B 1E 79 16]
Class 2 Request
Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:0] LinkAdr: 30

-> 17:46:26.343 [10 09 1E 27 16]
Data Not Available Response
Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:28.921 [10 7B 1E 99 16]
Class 2 Request
Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:1] LinkAdr: 30

-> 17:46:32.578 [10 09 1E 27 16]
Data Not Available Response
Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

<- 17:46:32.921 [10 5B 1E 79 16]
Class 2 Request
Link Control: User Data Class 2 [PRM:1 FCV:1 FCB:0] LinkAdr: 30

-> 17:46:34.031 [10 09 1E 27 16]

Data Not Available Response

Link Control: NACK - Requested Data not available [PRM:0 ACD:0 DFC:0] LinkAdr: 30

Appendix D

Use Case Textual Specification

This chapter presents the detailed specification and course of events of each technician's use case.

D.1 Use Case Textual Specification

Table D.1: Configure serial port parameters use case textual description

	Use Case: Configure serial port parameters
Code	UC14
Name	Configure serial port parameters
Description	The technician can interact with the Java application or the web interface to manually configure the parameters for the serial port connection
Actors	Technician
Flow of Events	<ul style="list-style-type: none">• The technician accesses the interface• The technician inputs the values• The technician saves the modifications if the values are correct
Pre-conditions	None
Post-conditions	The inserted values are loaded in the application

Table D.2: Configure server use case textual description

Use Case: Configure server	
Code	UC15
Name	Configure server
Description	The technician can interact with the Java application or the web interface to manually configure the client's server IP
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the interface • The technician inputs the values • The technician saves the modifications if the values are correct
Pre-conditions	None
Post-conditions	The inserted values are loaded in the application

Table D.3: Configure operation mode use case textual description

Use Case: Configure operation mode	
Code	UC16
Name	Configure operation mode
Description	The technician can change the application's operation mode according to the desired function
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the serial interface • The technician chooses the operation mode • The technician saves the modifications if the values are correct
Pre-conditions	None
Post-conditions	The application runs in the selected mode

Table D.4: Configure application's parameters use case textual description

Use Case: Configure application's parameters	
Code	UC17
Name	Configure application's parameters
Description	The technician can interact with the Java application or the web interface to manually configure the application's runtime parameters (waiting timeout, frame offset, etc)
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the interface • The technician inputs the values • The technician saves the modifications if the values are correct
Pre-conditions	None
Post-conditions	The inserted values are loaded in the application

Table D.5: Generate configuration file use case textual description

Use Case: Generate configuration file	
Code	UC18
Name	Generate configuration file
Description	The technician can interact with the Java application's interface to generate the configuration file with the inserted or default values
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the interface • The technician verifies the configuration values • The technician generates the configuration file
Pre-conditions	None
Post-conditions	The file is created in the modem's root

Table D.6: Configure connections use case textual description

	Use Case: Configure connections
Code	UC19
Name	Configure connections
Description	The technician can interact with the online interface to edit existing connections or create new ones
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the online interface • The technician configures the connections • The technician submits the changes
Pre-conditions	The devices involved in the connection must be registered
Post-conditions	The connection is changed or created

Table D.7: View modems' list and details use case textual description

	Use Case: View modems' list and details
Code	UC20
Name	View modems' list and details
Description	The technician can interact with the online interface to view the list of all the modems registered in the system and view each modem's details
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the interface • The technician accesses the modems' list • The technician selects a modem and sees its details
Pre-conditions	The modems must be registered in the system
Post-conditions	The information is displayed

Table D.8: View SIM cards' list and details use case textual description

Use Case: View SIM cards' list and details	
Code	UC21
Name	View SIM cards' list and details
Description	The technician can interact with the online interface to view the list of all the SIM cards registered in the system and view each card's details
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the online interface • The technician accesses the SIM cards' list • The technician selects a SIM card and sees its details
Pre-conditions	The SIM cards must be registered in the system
Post-conditions	The information is displayed

Table D.9: View connections use case textual description

Use Case: View connections	
Code	UC22
Name	View connections
Description	The technician can interact with the online interface to view the existing active connections, their geographic location, the details and the devices associated with each connection
Actors	Technician
Flow of Events	<ul style="list-style-type: none"> • The technician accesses the online interface • The technician accesses the connections' map • The technician selects a connection and sees its details
Pre-conditions	The connections and the devices associated to that connection must be registered in the system
Post-conditions	The information is displayed

Table D.10: Reset modem use case textual description

	Use Case: Reset modem
Code	UC23
Name	Reset modem
Description	The technician can access the online interface, choose a specific modem and remotely reset it
Actors	Technician
Flow of Events	<ul style="list-style-type: none">• The technician accesses the online interface• The technician accesses one modem's details• The technician resets the modem
Pre-conditions	The modem must be registered in the system and must be active
Post-conditions	The modem is rebooted

References

- [1] G. Clarke and D. Reynders. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.
- [2] Engin Ozdemir and Mevlut Karacor. Mobile phone based scada for industrial automation. *ISA Transactions* 45, no. 1, 2006.
- [3] Arash Shoarinejad. Communication protocols in substation automation and scada. *GE Energy Network Reliability Products and Services*, 2005.
- [4] J. Fitch and H. Y. Li. Challenges of scada protocol replacement and use of open communication standards. *10th IET International Conference on Developments in Power System Protection*, March 29 2010.
- [5] Paulo S. Motta Pires and Luiz Affonso H. G. Oliveira. Security aspects of scada and corporate network interconnection: An overview. *International Conference on Dependability of Computer Systems*, May 2006.
- [6] J. Stoupis S. Mohagheghi and Z. Wang. Communication protocols and networks for power systems - current status and future trends. *IEEE/PES Power Systems Conference and Exposition*, March 2009.
- [7] J. Makhija and LR Subramanyan. Comparison of protocols used in remote monitoring: Dnp 3.0, iec 870-5-101 and modbus. *Tech. Rep*, 2003.
- [8] R. Kalapatapu. Scada protocols and communication trends. *EXPO*, 2004.
- [9] Joaquin Luque Jaime Benjumea Gemma Sanchez, Isabel Gomez and Octavio Rivera. Using internet protocols to implement iec 60870-5 telecontrol functions. *IEEE Transactions on Power Delivery* 25, no. 1, 2010.
- [10] D. Kang and R.J. Robles. Compartmentalization of protocols in scada communication. *International Journal of Advanced Science and Technology Volume 8*, July 2009.
- [11] Enrique Dorrzoro David Oviedo Sergio Martin Jaime Benjumea Veronica Medina, Isabel Gomez and Gemma Sanchez. Iec-60870-5 application layer for an open and flexible remote unit. *35th Annual Conference of the IEEE Industrial Electronics Society*, November 2009.
- [12] Enrique Dorrzoro David Oviedo Sergio Martin Jaime Benjumea Veronica Medina, Isabel Gomez and Gemma Sanchez. Iec-60870-5 application layer over tcp/ip for an open and flexible remote unit. *IEEE International Symposium on Industrial Electronics*, July 2009.

- [13] Antonio Gomes Varela Pedro Gama and Wolf Freudenberg. Iec 60870-5-104 as a driver to evolution of substation and distribution automation at edp. *20th International Conference and Exhibition on Electricity Distribution*, June 8 2009.
- [14] Usha Communications Technology. General packet radio service, 2000. Technology White Paper.
- [15] R. J. Bates. *Gprs:General Packet Radio Service*. McGraw-Hill Professional, 2004.
- [16] M. Ismail M. Krishnan and K. Annuar. Radio resource and mobility management in gprs network. *Global Research and Development in Electrical and Electronics Engineering*, July 2002.
- [17] SIEMENS. Wireless module tc65, 2006. Siemens TC65 Datasheet.
- [18] Gsm/gprs modems and gsm/gprs modules. Available at <http://www.gsmfavorites.com/gsmhardware/>, last accessed on 04/06/2013.
- [19] Overview of gsm/gprs modems. Available at <http://www.smssolutions.net/hardware/gsmgprs1/>, last accessed on 04/06/2013.
- [20] Ge Liu Xingquan Xiao, Zhong Fu and Chuang Deng. A backup data network for power system automations based on satellite communication. *International Conference on Power System Technology: Technological Innovations Making Power Grid Smarter*, October 2010.
- [21] Sat runner. Available at <http://www.satrunner.com/en/satellite-phones-type/remote-satellite-modem-terminal.html>, last accessed on 04/06/2013.
- [22] Orbcomm global satellite communications. Available at <http://www.orbcomm.com/>, last accessed on 13/06/2013.
- [23] Thuraya. Available at <http://www.thuraya.com/>, last accessed on 13/06/2013.
- [24] Iridium satellite provider. Available at <http://www.iridium.com/default.aspx>, last accessed on 13/06/2013.
- [25] Inmarsat mobile satellite company. Available at <http://www.inmarsat.com/>, last accessed on 13/06/2013.
- [26] Globalstar, inc. Available at <https://la.globalstar.com/pg/?rls=1>, last accessed on 13/06/2013.
- [27] JSR 228 Expert Group. Information module profile next generation. Technical report, Siemens AG and Nokia Corporation, October 2005.
- [28] Direção de Tecnologia e Inovação. Instalações de telecomunicações - unidade remota de teleação e automatismos para subestação. Technical report, EDP Distribuição, January 2011.
- [29] Direção de Tecnologia e Inovação. Protocolos de comunicação de sistemas de proteção, comando e controlo numéricos (spcc). Technical report, EDP Distribuição, November 2011.
- [30] James Gosling. *Java 8482 Language Specification*. Sun Microsystems, 2000.
- [31] SIEMENS. Java user's guide, 2005. Siemens Cellular Engine Java Development Guide.

- [32] Jack Shirazi. *Java Performance Tuning*. O'Reilly Media Incorporated, 2003.
- [33] Nokia Corporation. Efficient midp programming, 2004.
- [34] Egon Börger and Robert F Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer Verlag, 2003.
- [35] Jon C Snader. *Effective Tcp/Ip Programming: 44 Tips to Improve Your Network Programs*. Addison-Wesley Professional, 2000.
- [36] Olaf Zimmermann Cesare Pautasso and Frank Leymann. Restful web services vs. big'web services: Making the right architectural decision. *17th international conference on World Wide Web*, 2008.
- [37] Michael Prior-Jones. Satellite communications systems buyers' guide. Technical report, British Antarctic Survey, 2009.
- [38] Jack Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge Academic, 1988.
- [39] Fan R. L. Cheded Fan and O. Toker. Internet-based scada: A new approach using java and xml. *Elektron 23, no. 1*, 2006.