Chapter 3 : SYSTEM OVERVIEW

This section provides an overview of the larger project in which this research area is a part of. It attempts to provide the reader with a clearer picture of the domain in which the design was developed and implemented.

3.1 Existing System

The IGUANA gateway is discussed in Chapter 2. The gateway has multiple server applications running on it. The server applications that were the focus of this project are openLDAP server, MySQL database server and ESD server.

The ESD server controls access to all the FAN daemons that provide access to each field area network. The current implementations of openLDAP and the SQL database have been adapted to query ESD to obtain information about a FAN and to update their directories or database respectively.

Each server has a separate client application that requests information about a particular field area network. The gateway will use some sort of security mechanism, either Secure Sockets Layer (SSL) or Transport Layer Security (TLS) to provide a secure connection for transporting messages between client and server.

The client applications will either use data existing on the gateway, for example, openLDAP will use data point information stored in a directory or it may send a query to ESD to obtain the information it requires.

The openLDAP and SQL implementations do not provide all the functionality as provided by ESD, such as REFRESHNODELIST, UPDATENODELIST etcetera. These commands can only be sent to the ESD server directly from the ESD client. ESD itself may use the event and log files on the gateway to respond to certain types of commands.

The figure below provides an overview of the components in the current system and their interactions.

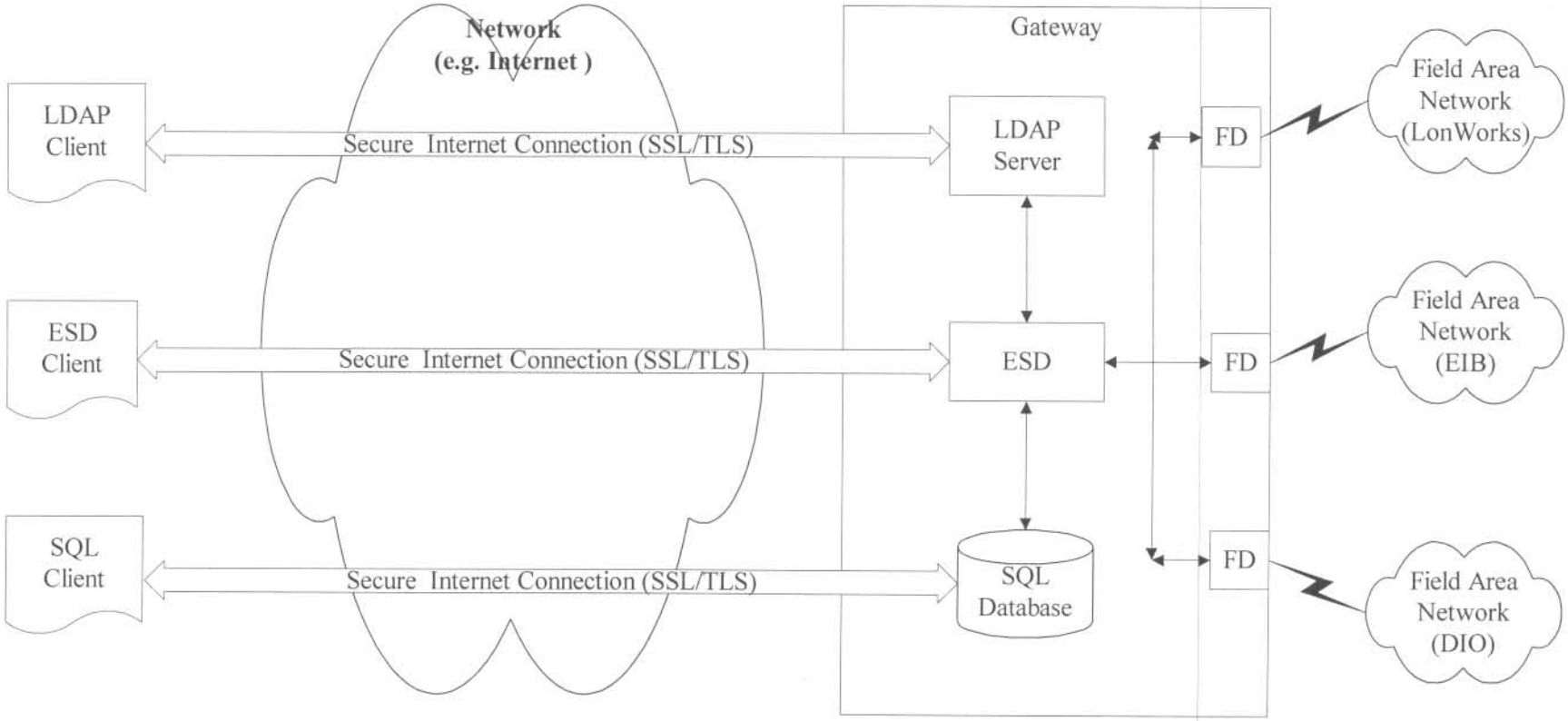

Figure 7: Current System Overview

Note: FD stands for Field Area Network (FAN) Daemon.

3.2 Proposed System

In the common XML messaging system, there is a common client interface to heterogeneous client-server applications that use network transport protocols such as TCP. The server specific messaging protocol is abstracted away from the client side, i.e. the client is provided with a generic user interface to the gateway irrespective of the protocol specific data access method on each server.

The client sends command messages to a common server (the XmlGateway), using a predefined XML format. The XmlGateway module uses the location of the information as a mechanism to determine which server on the gateway the message is intended for. The message is then reformatted into the specific server's message protocol format and sent to the server.

The response from the server is reformatted by the XmlGateway into an XML message response format as specified in a predefined XML document and sent to the client. All clients, therefore only have to process XML type requests and responses. This reduces the complexity of the client application and creates a truer thin client – fat server architecture.

Because messages are sent as an XML document, they can be transported over a number of transport protocols, such as HTTP, TCP/IP and encapsulated within a SOAP envelope. Therefore, if in the future, the gateway was placed behind a firewall, and the firewall restricted access to the HTTP port, the XML messages can still be transported via the firewall.

The XmlGateway currently provides access to LDAP, ESD servers and a SQL database. It is easily expandable and a different server application can be added with minimum overhead.

The proposed system using a common XML messaging protocol is shown in the figure below.

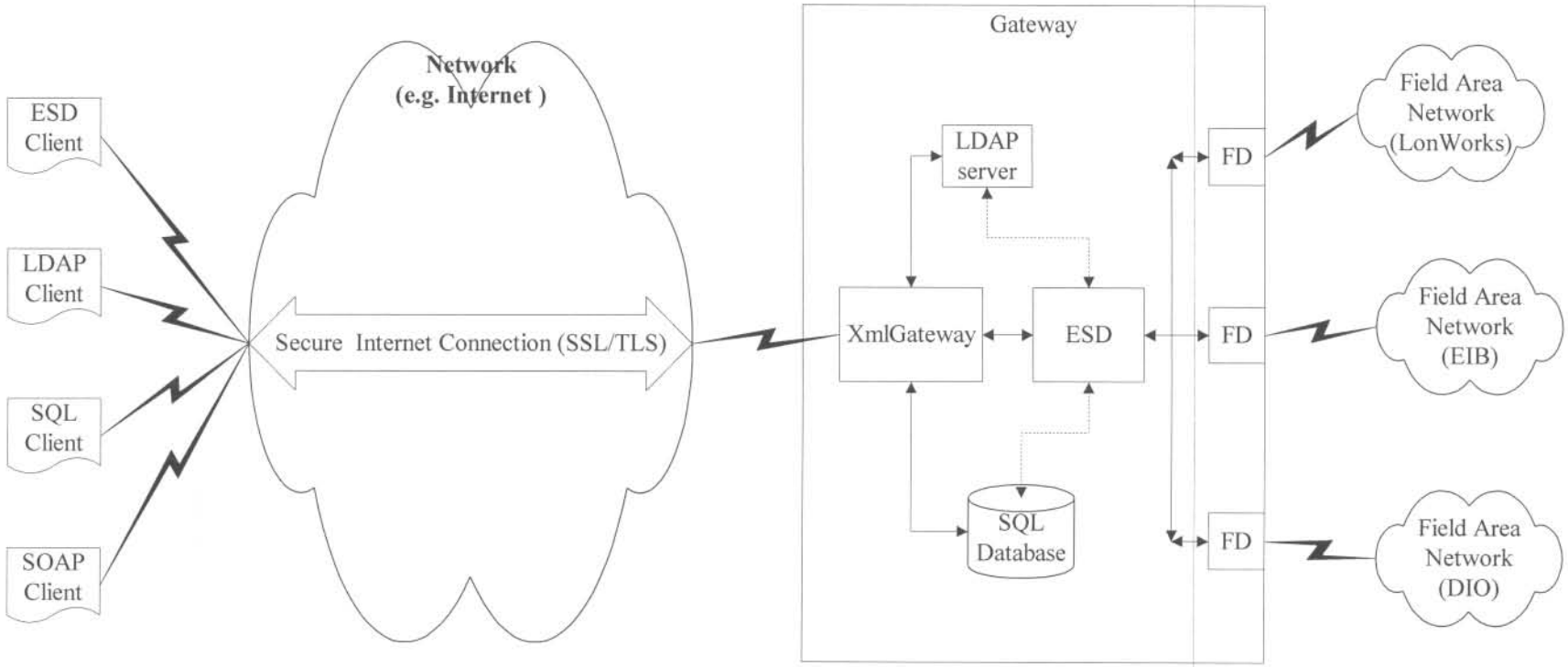

Figure 8: Overview of common messaging system using XML

Note: FD stands for Field Area Network (FAN) Daemon.

The slashed lines between LDAP, SQL and ESD indicate that the system can easily be integrated to the openLDAP and SQL implementations shown in the previous section, which do connect to ESD to obtain FAN data.

Chapter 4 : SYSTEM SPECIFICATIONS

This section describes the system requirements and functional specifications, the software and methodology used and the testing that was done to verify and validate the system.

In addition, a brief overview of the external software packages utilised in the implementation are provided.

4.1 Introduction

The IGUANA gateway currently provides access to data in a field area network node using ESD. It also runs versions of an LDAP server (openLDAP) and a database server that provides access to the same data.

The focus of this research to develop a system to send and receive messages to each of the above-mentioned applications using XML to represent the messages in a common format.

The following sections describe the system requirements identified, the system constraints, and the functional specification.

4.2 System Requirements

The following system requirements were identified:

1. Design and develop a component that allows clients to retrieve data from the IGUANA gateway using XML as the data description language
2. The system should provide access to ESD, LDAP and SQL
3. The component should interface with the above-mentioned applications to access data point and node information, event tables and logs.
4. The system should have some sort of web-based interface to provide online access.
5. An analysis of the performance (speed) for each of the above-mentioned applications should be carried out.

6. Prefer usage of Java as the programming language because it can be used across multiple platforms and does not require re-compilation on different operating systems
7. Prefer use of open source software to limit costs

4.3 System Constraints

The following system constraints were identified:

1. Use XML as the data description language.
2. Application must run on the Linux operating system.
3. There are limited resources available in terms of processor power and memory (486 processor and 8M RAM), but it can be upgradeable if required.

4.4 Assumptions

- The design will not be responsible for security of the gateway data or the security of messages sent and received over the Internet or fieldbus.
- The design will not be responsible for access control and authentication of user login information.
- The XML component may eventually make use of other sub-systems such as SSL or TLS to transmit data securely over the Internet.

4.5 Data Structure

The standard ESD schema is used for compatibility across all server implementations. The reason for this is to enable easier integration between the different application servers in the future and to ensure as close correlation as possible between the different application servers when analyzing the results.

For SQL and LDAP not all commands are implemented in the implementations currently available (i.e. from the IGUANA project). For the sake of completeness to obtain at least one complete comparison, it was decided to add a GENERAL table to the database design that would contain the information (such as eventlistseqno) that is not available in the current table

design. Refer to "Addendum A: IGUANA Structured Query Language Daemon (ISQLD)" for a description of the database schema.

It was felt that the database server application would be easier to change and remove the changes from at a later stage if the programs were integrated. However, the updatenodelist and refreshnodelist functionality that is specific to ESD is not implemented. The LDAP implementation only implements the objectclasses specified in the documentation [5]. Refer to "Addendum B: IGUANA LDAP Schema" for a description of the LDAP schema.

4.6 Functional Specification

The following diagram describes the main functional blocks in the project. The arrows indicate the direction of data flow. A description of each functional unit (FU) or user interface (IF) is provided after the diagram.

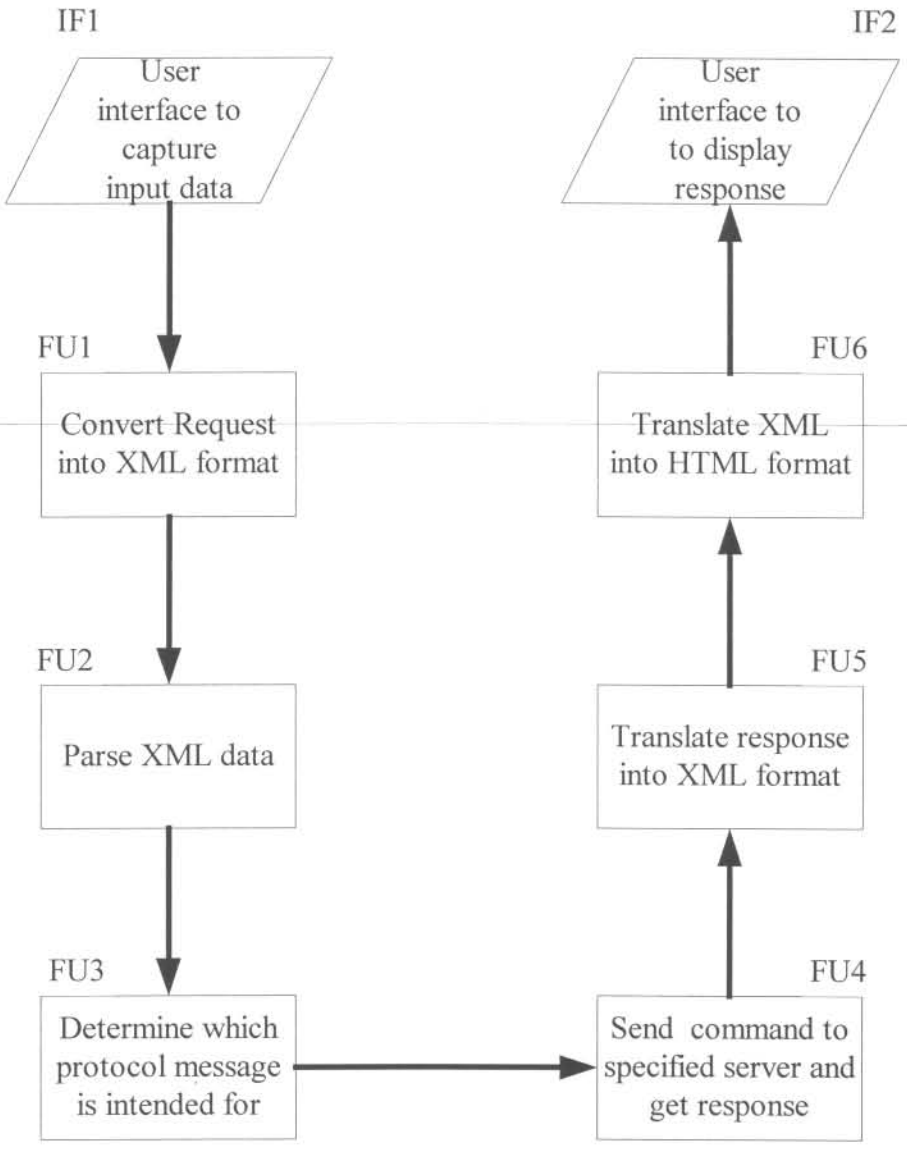

Figure 9: Functional block diagram of the proposed system

IF1: This is a web-browser type user interface. The screen displays forms to capture user input. There are separate forms for each command. It comprises Java Server Pages (JSPs) that display the forms used to capture user input per command.

FU1: This functional unit converts the HTTP request into the predefined XML document format.

FU2: This functional unit parses the XML data stream it receives and stores the information as element-value pairs.

FU3: This functional unit uses the location of the data to determine which application server the request should be sent to.

FU4: This functional unit connects to the specified application server and sends the request command to the server. It waits for a response from the server.

FU5: This functional unit converts the response received from the application server into the XML format as defined in the XML Schema.

FU6: This functional unit translates the XML response data stream into HTML format as specified by a selected stylesheet.

IF2: This user interface displays the response to the request sent earlier in HTML format.

The main design components are FU2, FU3, FU4 and FU5. They are concerned with processing an input XML data stream, sending a command to the correct application and processing the command response back into an XML data stream.

The units: FU1 and FU6 are peripheral and can easily be replaced if a client applications decides to send an XML data stream directly to FU2 and to receive an XML data stream back as a response.

The IF1 and IF2 and FU1 and FU6 blocks provide completeness in the system to show how it can be used with a browser based interface.

4.7 Software Methodology

The methodology followed was an iterative software development lifecycle methodology using a feedback based waterfall method. This means that after completing a needs analysis by determining the system requirements and system constraints, an initial system architecture and high-level design was developed. This architecture and design was then discussed with the relevant stakeholders and where applicable, the design was reworked.

The design was then implemented for the ESD application only using the Java programming language on the Linux (Red Hat) operating system. The design and XML model was again refined as implementation progressed. Each section of implementation was initially developed as independent entities and tested individually. The final system was integrated to provide a complete application that consisted of a user interface and backend processes working together.

The system was shown to relevant stakeholders and changes made. Because the system is designed to enable new client-server applications to be added on with minimum additional coding time required, the MySQL and openLDAP implementation were then developed and tested.

4.8 Testing

The testing of the system was carried out on a micro and macro level. Initially, a simple test program was built to ensure that a connection to the server application could be established and messages processed.

Then, separate test programs were developed for each server implementation which verified that XML and non-XML messages are correctly processed when sent to the specified server and that the respective XML or non-XML response messages were received.

For the front-end system, a proof of concept test program was written to test if an XSLT program would correctly transform XML into HTML format.

Finally the entire system was integrated and the actual application was tested from user input to backend server applications, through to the conversion of responses into HTML format.

4.9 External Software Components

The following section lists the external software applications that were used and provides a brief description of each application.

The software components used are all open source applications. Open source means that it is possible for anyone to use and modify the code, as it is freely available (generally for non-commercial purposes).

4.9.1 Database

The database used is MySQL. MySQL is a multi-user, multi-threaded, relational database management system. Clients may connect to the MySQL server using sockets or named pipes. There also are ODBC and JDBC drivers available that allow application programs to connect to MySQL (refer to www.mysql.com for more information). The database was chosen for the following reasons:

- It is an open source application.
- It is a fast, reliable, easy to use relational database system.
- It supports ANSI SQL.

4.9.2 Directory Server

The directory server used is OpenLDAP. The OpenLDAP program was originally developed as a project at the University of Michigan. Further and future development is now handled by the OpenLDAP foundation. Refer to www.openldap.org for more information and specifically to references [24, 25, 27]. The OpenLDAP application was chosen for the following reasons:

- It is an open source application.
- The OpenLDAP implementation supports the complete LDAP functionality needed for setting up an LDAP service on a Linux machine.
- It conforms to the LDAP standards.
- It is the LDAP service currently used in the IGUANA project.

4.9.3 Web Server

The web server user is Tomcat (version 4). Tomcat is a Java Servlet and JSP container developed by The Apache Software Foundation (www.apache.org). Tomcat was chosen for the following reasons:

- It is an open source application.
- It is relatively easy to configure and use.
- It is reliable and stable and supports usage of JSP.
- It is part of the well known and widely used and supported Apache suite of enterprise products.

4.9.4 XML Parser

The XML parser used is Xerces. Xerces is a reliable and easy to use tool for XML parsing and generation. Xerces was chosen for the following reasons:

- It is an open source application.
- It is relatively easy to configure and use.
- It is available for both Java and C++.
- It implements the W3C XML and DOM (Level 1 and 2) standards, as well as the de facto SAX (version 2) standard.
- The parsers are highly modular and configurable.
- It provides initial support for XML Schema (draft W3C standard).
- It is part of the well known and widely used and supported Apache suite of enterprise products.

4.9.5 XML Translator

The XML translator used is Xalan. Xalan provides high-performance XSLT stylesheet processing. The reasons for using Xalan are:

- It is an open source application.
- It is relatively easy to configure and use.
- It is available for both Java and C++.
- It implements the W3C XSLT and XPath recommendations.
- It is part of the well known and widely used and supported Apache suite of enterprise products.

4.9.6 Programming Language

The programming language used is Java (version 1.4). The reasons for using Java are:

- It is a relatively open standard programming language.
- It can be used across multiple platforms.
- It supports object-oriented programming.

4.9.7 Operating system

The operating system used is Red Hat Linux. The reasons for using Linux are:

- It is an open source application.
- It is relatively easy to use.
- All the above-mentioned software applications are compiled to work on this operating system.