Understanding and Simulating the IEC 61850 Standard *

Yingyi Liang Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{*yliang6, rhc*}*@uiuc.edu*

Abstract

The recent international standard IEC 61850 proposes a unified solution of the communication aspect of substation automation. However, the standard itself is not easily understood by users other than domain experts. We present our understanding of the IEC 61850 standard as well as the design and implementation of our simulation tool in this report. Also, we give suggestions on the implementation of this standard based on our experience and lessons in the development of our simulation.

1 Introduction

Today, power substations are mostly managed by substation automation systems. These systems employ computers and domain specific applications to optimize the management of substation equipment and to enhance operation and maintenance efficiencies with minimal human intervention [8].

Once upon a time, substation automation systems utilized simple, straightforward and highly specialized communication protocols [7]. These protocols concerned less about the semantics of the exchanged data, data types of which were relatively primitive. Equipment was dumb and systems were simple. However, today's substation automation systems can no longer enjoy such simplicity because of their growing complexity — equipment becomes more intelligent and most of those simple old systems have been gradually replaced by open systems, which embrace the advantage of emerging technology like relational database systems, multi-task operating systems and support for state-of-the-art graphical display technology.

Besides, devices from different manufacturers used different substation automation protocols [9, 3, 12], disabling them to talk to each other. Utilities have been paying enormous money and time to configure these devices to work together in a single substation. Today most utilities and device manufacturers have recognized the need for a unified international standard to support seamless cooperation among products from different vendors.

The IEC 61850 international standard, drafted by substation automation domain experts from 22 countries, seeks to tackle the aforementioned situation. This standard takes advantage of a comprehensive object-oriented data model and the Ethernet technology, bringing in great reduction of the configuration and maintenance cost. Unlike its predecessor, the Utility Communication Architecture protocol 2.0 (UCA 2.0) [12], the IEC 61850 standard is designed to be capable for domains besides substation automation. To make the new protocol less domain dependent, the standard committee endeavored to emphasize on the data semantics, carving out most of the communication details. This effort, however, could result in difficulties in understanding the standard.

In this research project, we aim to get a clear understanding of the IEC 61850 standard and simulate the protocol based on J-Sim [11]. Our ultimate goal is to investigate the security aspect about the IEC 61850 standard.

2 The IEC 61850 standard

The first release of the IEC 61850 consists of a set of documents of over 1,400 pages. These documents are divided into 10 parts, as listed in Table 1. Part 1 to Part 3 give some general ideas about the standard. Part 4 defines the project and management requirements in an IEC 61850 enabled substation. Part 5 specifies the required parameters for physical implementation. Part 6 defines an XML based language for IED configuration, presenting a formal view of the concepts in the standard. Part 7 elaborates on the logical concepts, which is further divided into four subparts (listed in Table 2). Part 8 talks about how to map the internal objects to the presentation layer and to the Ethernet link layer. Part 9 defines the mapping from sampled measurement value (SMV)

*This work was funded by NSF CNS 03-05537.

to point-to-point Ethernet. The last part gives instructions on conformance testing. Since Part 7 defines the core concepts of the IEC 61850 standard, we will focus on this part in this report.

Subpart	Title
7-1	Principles and Models
7-2	Abstract Communication Service Interface
7-3	Common Data Classes
7-4	Compatible Logical Node Classes and Data Classes

Table 2: Subparts of IEC 61850-7

The IEC 61850 standard is not easy to understand for people other than experts in the substation automation domain due to the complexity of the documents and the assumed domain-specific knowledge. Introductory documents on the standard abound [13, 4, 7, 5, 8, 2], but most of them are in the view of substation automation domain experts. Kostic et al. explained the difficulties they had in understanding the IEC 61850 standard [7]. In this section, we provide another experience of understanding this standard, trying to explain the major concepts of the IEC 61850 standard.

2.1 Challenges

Understanding the IEC 61850 standard proposes the following challenges for a outsider of the substation automation domain:

1. As a substation automation standard proposed by a group of domain experts, the IEC 61850 protocol assumes quite an amount of domain-specific knowledge, which is hardly accessible by engineers and researchers out of the substation automation domain. To make things worse, the terms used in the standard is to some extent different from those commonly used in software engineering, bringing some difficulties for software engineers in reading the standard.

2. The entire standard, except Part 6, is described in natural language with tables and pictures, which is known to be ambiguous and lack of preciseness. This situation is problematic because the IEC 61850 concepts are defined by more than 150 mutually relevant tables distributed over more than 1,000 pages. A formal presentation of all these concepts would be appreciated.

3. The experts proposing this protocol come from 22 different countries and are divided into 10 working groups, each responsible to one part of the standard. Due to the different backgrounds and the informal presentation style of the standard, the standard contains a considerable number of inconsistencies. Such inconsistencies are more obvious for different parts of the standard, e.g. the data model described in Part 6 is clearly different from that described in Part 7.

4. The standard committee made a great effort to describe the protocol in an object-oriented manner but the result is not so object-oriented. For example, the ACSI services are grouped by different classes, but reference to the callee object is not defined as a mandatory argument of the service function.

5. The standard is designed to be implementation independent but this is not always true. For example, the data attribute `TimeAccuracy` in Part 7-2 Table 8 is defined as `CODED ENUM`, while what it virtually represents is a 5-bit unsigned integer; the frequent use of `PACKED LIST` (i.e. "bit fields" in the C language) also brings implementation details to interface design.

6. Things are mixed up in the documents. Mandatory components and optional components are mixed in the standard, and domain independent concepts are mixed up with domain specific concepts. Even though the optional components and mandatory ones are marked with "O" and "M" alternatively, it would be a tough task to refine a model consisting only the mandatory components due to the implicit dependences between attributes in different tables and the conditional inclusion of some attributes. In fact, there are 29 common data classes and 89 compatible logical nodes defined in the standard, the relationship among which is unclear.

2.2 Intelligent electronic device

In the past, utility communication standards usually assumed some domain-specific background of the readers. Consequently, they contained a lot of implicit domain knowledge, which is hardly accessible to outsiders (e.g. software engineers) [7]. The IEC 61850 standard does not escape from this category. To help understanding the logical concepts of IEC 61850, we would like to lay a basic idea of intelligent electronic devices (IED), the essential physical object hosting all the logical objects.

Basically, the term intelligent electronic device refers to microprocessor-based controllers of power system equipment, which is capable to receive or send data/control from or to an external source [8]. An IED is usually equipped with one or more microprocessors, mem-

Part	Title
1	Introduction and Overview
2	Glossary
3	General Requirements
4	System and Project Management
5	Communication Requirements for Functions and Device Models
6	Configuration Description Language for Communication in Electronic Substations Related to IEDs
7	Basic Communication Structure for Substation and Feeder Equipment
8	Specific Communication Service Mapping (to MMS and to Ethernet)
9	Specific Communication Service Mapping (from Sampled Values)
10	Conformance Testing

Table 1: Parts of the IEC 61850 standard documents

ory, possibly a hard disk and a collection of communication interfaces (e.g. USB ports, serial ports, Ethernet interfaces), which implies that it is essentially a computer as those for everyday use. However, IEDs may contain some specific digital logics for domain-specific processing.

IEDs can be classified by their functions. Common types of IEDs include relay devices, circuit breaker controllers, recloser controllers, voltage regulators etc.. It should be noted that one IED can perform more than one functions, taking advantage of its general-purpose microprocessors. An IED may have an operating system like Linux running in it.

2.3 Substation architecture

A typical substation architecture is shown in Figure 1. The substation network is connected to the outside wide area network via a secure gateway. Outside remote operators and control centers can use the abstract communication service interface (ACSI) defined in Part 7-2 to query and control devices in the substation. There is one or more substation buses connecting all the IEDs inside a substation. A substation bus is realized as a medium bandwidth Ethernet network, which carries all ACSI requests/responses and generic substation events messages (GSE, including GOOSE and GSSE). There is another kind of bus called process bus for communication inside each bay. A process bus connects the IEDs to the traditional dumb devices (merge units, etc.) and is realized as a high bandwidth Ethernet network. A substation usually has only one global substation bus but multiple process buses, one for each bay.

ACSI requests/responses, GSE messges and sampled analog values are the three major kinds of data active in the substation network. Since we are less interested in communication on the process buses (like sampled value multicasting), we focus on the activities on the substation bus in this report, especially the ACSI activities.

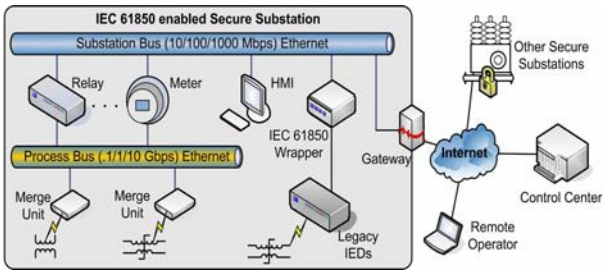

Figure 1: Substation architecture

Interactions inside a substation automation system mainly fall into three categories: data gathering/setting, data monitoring/reporting and event logging. The former two kinds of interactions are the most important — in the IEC 61850 standard all inquiries and control activities towards physical devices are modeled as getting or setting the values of the corresponding data attributes, while data monitoring/reporting provides an efficient way to track the system status, so that control commands can be issued in a timely manner.

To realize the above kinds of interaction, the IEC 61850 standard defines a relatively complicated communication structure, as is shown in Figure 2. Five kinds of communication profiles are defined in the standard: the abstract communication service interface profile (ACSI), the generic object oriented substation event profile (GOOSE), the generic substation status event profile (GSSE), the sampled measured value multicast profile (SMV), and the time synchronization profile. ACSI services enable client-server style interaction between applications and servers. GOOSE provides a fast way of data exchange on the substation bus and GSSE provides an express way of substation level status exchange. Sample measured value multicast provides an effective way to exchange data on a process bus.

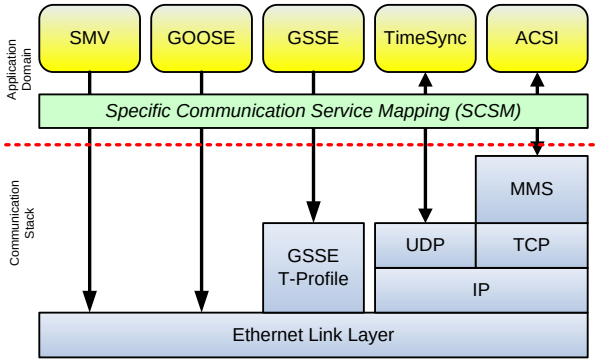

Figure 2: The communication profiles

2.4 Abstract communication service interface

ACSI is the primary interface in the IEC 61850 standard not only because it is the interface via which applications talk with servers, but also in the sense that the ACSI communication channel is an important part of a logical connection between two logical nodes. ACSI defines the semantics of the data exchanged between applications and servers, thus it becomes the major part of the IEC 61850 standard.

The standard committee adopt an object-oriented approach in the design of ACSI, which includes a hierarchical and comprehensive data model and a set of available services for each class in this data model. Although the data model is usually described outside the scope of the ACSI, it is actually part of it. The benefits of using an object-oriented utility communication interface are two fold. On the one hand, objects (e.g. registers) can be referenced in an intuitive way (e.g. "`Relay0/MMXU0.voltage`") instead of by the traditional physical address (like `Reg#02432`). On the other hand, software engineers can build more reliable applications using such service interface.

In the following two sections, we present a brief description on these two ACSI components.

2.5 Data model

The hierarchical data model defined in the IEC 61850 is depicted in Figure 3 and Figure 4.

Server is the topmost component in this hierarchy. It serves as the joint point of physical devices and logical objects. Theoretically one IED may host one or more server instances, but in practice usually only one server instance runs in an IED. A server instance is basically a program running in an IED, which shares the same meaning with other servers like FTP server etc.. Each server has one or more access points, which are the logical representation of a NIC. When a client is to access data or service of the server, it should connect to an access point of this server and establish a valid association.

Each server hosts several files or logical devices. Clients can manipulate files in the server like talking to a FTP server, which is usually used as a means to upload/update the configuration file of an IED. A logical device is the logical correspondence of a physical device. It is basically a group of logical nodes performing similar functions.

Functions supported by an IED are conceptually represented by a collection of primitive, atomic functional building blocks called logical nodes. The IEC 61850 standard predefines a collection of template logical nodes (i.e. compatible logical nodes) in Part 7-4. Besides the regular logical nodes for functions, the standard also requires every logical device have two specific logical nodes: Logical Node Zero (LN0) and LPHD, which correspond to the logical device and the physical device, alternatively. Besides holding status information of the logical device, LN0 also provides additional functions like setting-group control, GSE control, sampled value control etc.. In the IEC 61850 standard, the entire substation system is modeled as a distributed system consisting of a collection of interacting logical nodes, which are logically connected by logical connections. It should be noted that the term logical connection refers to the logical concept of the connections between two logical nodes, which can be direct or indirect or even a combination of many different kinds of communication channels. In fact, the connection of two logical nodes is usually both indirect and a combination of TCP, UDP and direct Ethernet connections. We will explain logical connections in Section 2.9.

Data exchanged between logical nodes are modeled as data objects. A logical node usually contains several data objects. Each data object is an instance of the DATA class and has a common data class type. Similar to the concept of objects in most object-oriented programming languages, a data object consists of many data attributes, which are instances of data attributes of the corresponding common data class. Data attributes are typed and restricted by some functional constraints. Instead of grouping data attributes by data objects, functional constraints provide a way to organize *all* the data attributes in a logical node by functions. Types of data attributes can be either basic or composite. Basic types are primitive types in many programming languages, whereas composite types are composition of a collection of primitive types or composite types.

In the IEC 61850 standard, data attributes are at least as important as, if not more than, data objects for two

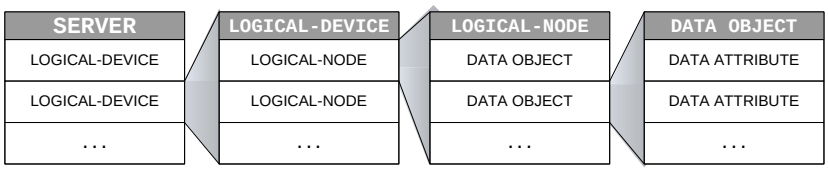

Figure 3: Hierarchy of the IEC 61850 data model

reasons. Firstly, data objects are just logical collections of the contained data attributes while (primitive) data attributes are the de facto logical correspondence to the physical entities (memory units, registers, communication ports, etc.); secondly, the purpose of data objects is for the convenience of managing and exchanging values of a group of data attributes sharing the same function.

Despite data objects, the IEC 61850 standard provides the concept of data set as another ways to manage and exchange a group of data attributes. Members of a data set can be data objects or data attributes. The concept of data set is somewhat similar to the concept of view in the area of database management systems. In the IEC 61850 standard, most services involve data sets. Members in a data set unnecessarily come from the same logical node or the same data object, thus providing high flexibility of data management. Data sets are categorized into permanent ones and temporary ones. Permanent data sets are hosted by logical nodes and will not be automatically deleted unless on the owners' explicit requests; temporary data sets are exclusively hosted by the association having created them and will be automatically deleted when the association ends.

2.6 Service model

Services provided by ACSI include querying object set, getting/setting data values, controlling system objects, report manipulation, log manipulation, and other services like file upload/download. Table 3 gives a list of ACSI services defined in the IEC 61850 standard.

All ACSI services are requested by applications and responded by servers. In order to request a service in a server, an application must first establish a valid two-party application association (TPAA) with the server. The TPAA maintains the session states and provides a virtual view of the server to the application. A typical interaction procedure between an application A and a server S goes as follows:

1. A establishes a TCP connection with S;
2. A "logs in" to S by requesting the `Associate` service from S, providing authentication related information as parameters;

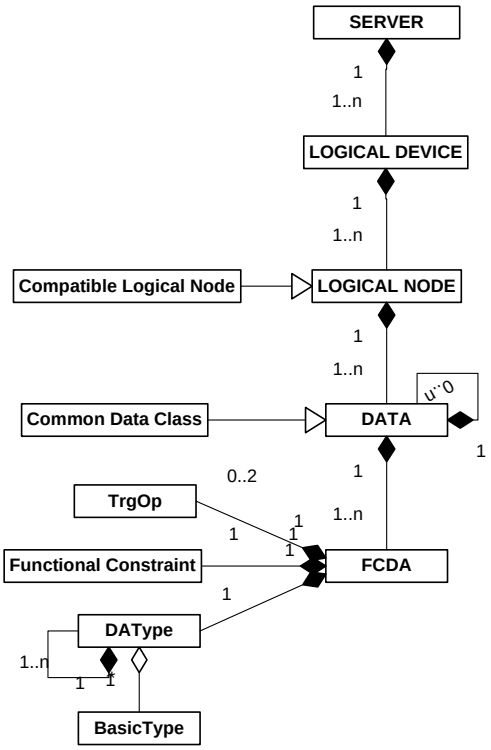

Figure 4: The data model of the IEC 61850

3. S validates the information provided by A and creates a TPAA object, which provides a virtual view of S to A;
4. A requests subsequent services while S processes the requests and responses with appropriate responses defined in the IEC 61850 standard;
5. A issues a `Release` request to S;
6. S reclaims the TPAA of A and ends the session.

The virtual view of server provided by a TPAA enforces the access control policies set forth by the server. This virtual view defines which objects in the server are visible and accessible to the application and what kinds of service of those objects are accessible from the application. The concept of virtual view is very flexible and

SERVER:	**LOG-CONTROL-BLOCK:**
GetServerDirectory	GetLCBValues
ASSOCIATION:	SetLCBValues
Associate	QueryLogByTime
Abort	QueryLogAfter
Release	GetLogStatusValues
LOGICAL-DEVICE:	**GOOSE:**
GetLogicalDeviceDirectory	SendGOOSEMessage
LOGICAL-NODE:	GetGoReference
GetLogicalNodeDirectory	GetGOOSEElementNumber
GetAllDataValues	GetGoCBValues
DATA:	SetGoCBValues
GetDataValues	**GSSE:**
SetDataValues	SendGSSEMessage
GetDataDirectory	GetGsReference
GetDataDefinition	GetGSSEDataOffset
DATA-SET:	GetGsCBValues
GetDataSetValues	SetGsCBValues
SetDataSetValues	**MULTICAST-SAMPLE-VALUE-CONTROL-BLOCK:**
CreateDataSet	SendMSVMessage
DeleteDataSet	GetMSVCBValues
GetDataSetDirectory	SetMSVCBValues
Substitution:	**UNICAST-SAMPLE-VALUE-CONTROL-BLOCK:**
SetDataValues	SendUSVMessage
GetDataValues	GetUSVCBValues
SETTING-GROUP-CONTROL-BLOCK:	SetUSVCBValues
SelectActiveSG	**Control:**
SelectEditSG	Select
SetSGValues	SelectWithValue
ConfirmEditSGValues	Cancel
GetSGValues	Operate
GetSGCBValues	CommandTermination
BUFFERED-REPORT-CONTROL-BLOCK:	TimeActivatedOperate
Report	**Time synchronization:**
GetBRCBValues	TimeSynchronization
SetBRCBValues	**FILE transfer:**
UNBUFFERED-REPORT-CONTROL-BLOCK:	GetFile
Report	SetFile
GetURCBValues	DeleteFile
SetURCBValues	GetFileAttributeValues

Table 3: ACSI services

the IEC 61850 standard does not place any restriction on the access control policies of the server. One possible and relatively simple access control is the world-group-owner access control for files used in many UNIX systems.

The ACSI interface defines an *object-oriented* interface for the applications but it does not require the internal implementation to be object-oriented. In actual fact, according to our experience on simulating the IEC 61850 protocol, object-oriented approach might not be a wise choice for the internal implementation.

2.7 Reporting and logging

The IEC 61850 standard provides an efficient mechanism called reporting for applications to track changes to the subscribed system objects. Instead of polling the data attribute values periodically, applications can group the interesting data attributes into a data set, and require the logical node hosting this data set report any changes to

the members of this data set. Theoretically a data set can contain data objects/data attributes from different logical nodes, but data sets for reporting usually contain only the data objects/data attributes in the same logical node. The procedure of report generation and transmission is under the control of an information block called report control block (RCB). A RCB maintains the necessary information to generate a report like which fields should be included in the report, on what events a report should be generated, the sequence number of the current report, whether this RCB is enabled, etc..

A typical report generation and transmission procedure is described as follows:

1. client application creates a data set containing all the data objects and data attributes it concerns;
2. client sets the parameters in a RCB, specifies the aforementioned data set as the source of the reports (this step is called to "subscribe to a data set") and enables this RCB;
3. on any change to any member of the data set, the logical node tests this change against the event list of the RCB, and issues an internal event if any event in the list gets matched;
4. on receiving the internal event, the RCB stores this event for later sending;
5. if the condition of sending the report is satisfied, the logical node collects necessary data, generates the report and sends it out to the client via the association under the direction of the parameters in the RCB.

Reports can be sent by either two-party application association (TPAA) or multi-party application association (MPAA). TPAA is the association that can serve only one client while MPAA can serve more than one client simultaneously. Reports generated on the request of the `Report` service is sent by TPAA, whereas other reports are sent by MPAA. Reporting uses a publisher/subscriber mechanism: for every client, the server must assign an individual RCB to handle the report generation and transmission.

Logging is a mechanism to record the device events. Logs are stored in the server and hosted by the corresponding logical device. Unlike reporting, every time a device event is triggered, the logical device merely saves a log entry into the log database for later inquiries.

2.8 Generic substation event

Besides reporting, the IEC 61850 standard defines generic substation event (GSE) as another means for applications to monitor changes to the data objects/data attributes. GSE is designed for fast delivering notification of system object changes. There are two kinds of GSE, generic object-oriented substation events (GOOSE) and generic substation state event (GSSE). GOOSE is used to exchange a wide range of common data while GSSE is used to convey state change information. The GSE mechanism shares a lot of similarities with reporting, with the major difference that GSE is designed for fast information exchange inside a substation while reporting is mainly used for sending notification from the server side to remote control centers or browsers.

Since real-time performance is critical for GSE messages, the message format and communication stack for GSE transmission is very different from those for reporting. GSE messages are transmitted in binary format, which provides shorter message body and higher message encoding/decoding speed. In stead of using TCP or UDP as the underlying transport layer, GSSE uses its specific transport layer while GOOSE messages are sent to the Ethernet link layer directly without going through any transport layer or network layer.

GSE also utilizes a publisher/subscriber mechanism to transmit the messages. This mechanism is implemented by the Ethernet multicast feature: the publisher sends the GSE message to a specific multi-cast MAC address and the subscribers pick up messages sent to this address, put them into their local buffer for the local applications to consume.

2.9 Communication network

The IEC 61850 standard defines a distributed system consisting of interacting logical nodes, which are connected by logical connections. However, in order for this distributed system to work correctly and intelligently, there must be some intelligent components inside this network. It is not hard to see that applications play this role. An interesting question is how to integrate applications into the interacting logical node network. We try to answer this question by starting from the simple ACSI server-application network.

Figure 5 shows such a server-application network. Clearly, each server may serve several applications and vice versa. The dotted lines in Figure 5 refer to the communication channels for reports and GSE messages. Although there are no restrictions about the relationship between report subscribers and report publishers, it is not true for GSE messages. The design of GSE has physical concerns. GSE messages are sent from one IED to another, thus the subscribers and the publishers should not reside in the same IED.

To make things more clear, let us transform Figure 5 into Figure 6, which shows the communication between

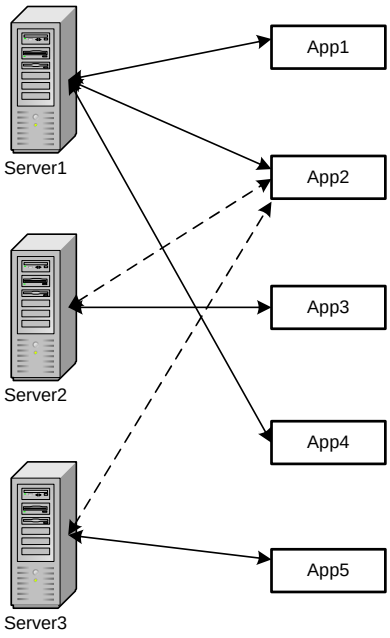

Figure 5: Server-Application network

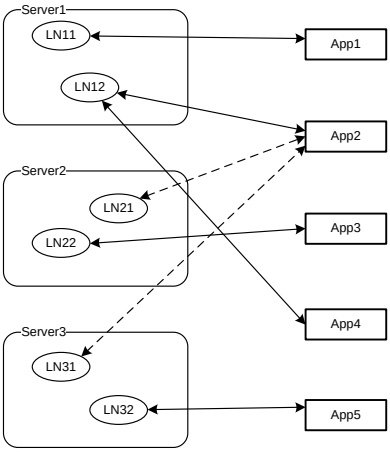

Figure 6: Logical node-application network

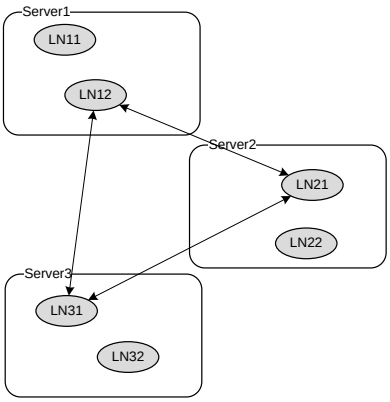

Figure 7: Logical node network

applications and logical nodes (we hid the communication to server and logical devices). At this point, we can clearly see that logical connections between logical nodes are actually a mixture of several kinds of connections: when a logical node sends a message to another logical node, it is virtually sending reports/GSE messages to the relevant applications; after necessary processing of the reports/GSE messages, the applications issue relevant requests to the other logical node, and vice versa. Thus we can deploy the application logic to the relevant logical nodes and get Figure 7 by abstracting away the applications. Being aware of this abstraction is important for both implementation and simulation of the IEC 61850 protocol.

3 Simulating the IEC 61850

In this section, we talk about the design and implementation of a simulator of the IEC 61850 standard. Due to the challenges listed in Section 2.1, simulating the entire protocol is very difficult. Since the goal of our simulation is to inspect possible security vulnerabilities in the protocol, we refined the protocol to a version containing only data gathering/setting related ACSI services and reporting services. Furthermore we simplified the data model by abstracting various data attribute types to string and discarding the concept of functional constraint. Currently, features of the IEC 61850 protocol supported by our simulation model include two-party application association, data attribute, data object, data set, logical node, logical device, server, ACSI services, reporting and unbuffered report control block.

3.1 Design and implementation of the simulator

In order to inspect the IEC 61850 standard, we need a tool to simulate the protocol. Demo software of the protocol exists [10]. However, they are mostly provided by the IED manufacturer and are not freely available for research purposes. Kostic et al. proposed an implementation of ACSI [6]. The work of Kostic et al. focuses on implementing a set of application programming interface for device and application development, while we would like to build our own simulation tool for further study on the IEC 61850, especially on network security issues.

Our simulator consists of about 3,500 lines of Java code. Its components are divided into three major categories: the data model, internal message representation and the service model. Figure 8 shows the architecture of the simulator.

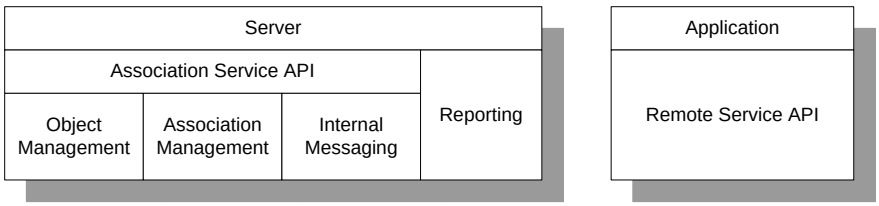

Figure 8: The IEC 61850 simulator architecture

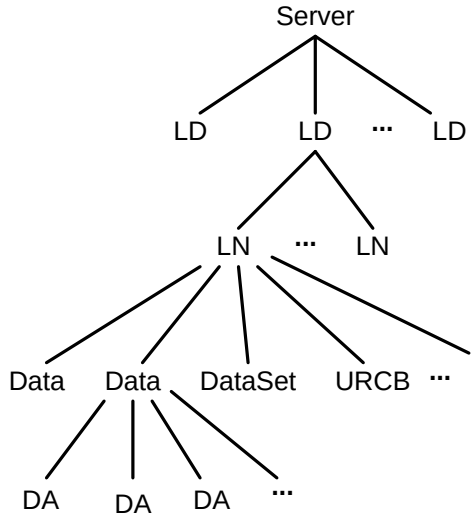

Figure 9: The IEC 61850 simulator data model

3.1.1 Data model

The internal data model in the simulator is the same as the one defined in the IEC 61850 standard. All the IEC 61850 objects are instances of the `I6Object` class and are organized as a tree as shown in Figure 9. `I6Object` class provides a collection of services to manage the objects. Table 4 lists some frequently used services.

Name	Description
`addChild`	adds a child node
`removeChild`	removes a child node
`getName`	gets the node name
`getRef`	gets the object reference
`create`	creates a node in the tree
`remove`	removes a node from the tree
`lookup`	looks up a node in the tree
`list`	lists the specific child nodes
`lock`	locks a specific node
`unlock`	unlocks a specific node

Table 4: Major services of `I6Object`

The `lock` and `unlock` services are necessary because the IEC 61850 standard allows multiple accesses to the same object simultaneously. One would resolve this problem by sequentializing all the received requests, but since reports could be generated at the time when the related data objects are being updated, synchronization mechanism is still needed. One could implement `lock` as locking the whole object tree, but this would lead to severe performance problems, even dead locks. Our solution is to lock the whole subtree dominated by the locked object and mark all the ancestor nodes of that object as partly locked so that these ancestor nodes cannot be locked by other Associations.

Each kind of the IEC 61850 objects are implemented as a subclass of `I6Object`. These subclasses provide templates to define the desired specialized classes. For example, in order to create a specialized logical device class, one just need to derive a subclass from the `LDevice` class and use the `declareLN` method to declare the member logical nodes in the `configure` function.

Using the IEC 61850 data model as the internal data model has two advantages: on the one hand, there is no need to map the internal data representation to the IEC 61850 data model in such an approach; on the other hand, we can deploy the implementation of the ACSI services to the relevant object classes thus achieve a simple object-oriented approach. However, our simulation experience shows such a simple internal data representation might not be a good choice because of the following two reasons:

1. The data model defined by the IEC 61850 is unnecessarily structured as a tree. Functional constraints present another view of data attributes in a logical node, breaking the tree structure maintained by data objects.

2. Some ACSI services are not easy to deploy to the specific object classes. One example is the `CreateDataSet` service, which should be implemented by the `LogicalNode` class or the `Association` class instead of the `DataSet` class.

To maintain the tree structure, one can implement functional constraints as implicit data sets, and redirect

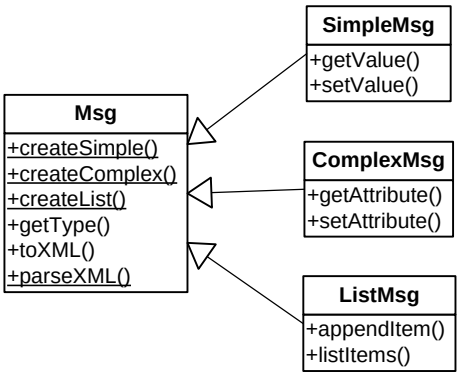

Figure 10: Internal message of the simulator

the service requests to the appropriate object class using a dispatcher in the `Association` class. Instead of maintaining an internal representation, we recommend an other approach: using a lightweight database system as the backend storage support. This approach provides several benefits:

1. A database system provides the most flexible and easiest way to maintain the appropriate tables and views. A database system incorporates all the necessary services to operate on the object tables, reducing the complexity of maintaining an internal data representation.

2. A database system supports exclusive object accesses, so the engineer does not need to explicitly perform lock and unlock operations.

3. Database systems are usually optimized for data storage and access, thus the database system approach can give comparable real-time performance against the internal data representation approach.

3.1.2 Message representation

Messages are used in two cases: as internal events and as requests/responses of ACSI services. Although the IEC 61850 standard defines manufacturing message specification (MMS) as its representation format for information exchange, we would like to use another information representation format because on the one hand, MMS is relatively complicated and on the other hand, the details about the MMS standard is not freely available.

Messages in our simulator are represented as attributed trees. There are three kinds of tree nodes (Figure 10): `SimpleMsg`, `ComplexMsg` and `ListMsg`. A `SimpleMsg` node is a leaf node containing a string value, which is used to represent a value. A `ComplexMsg` node is a node containing a collection of named attributes, the values of which are also nodes. A

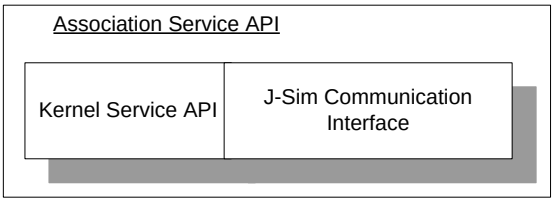

Figure 11: Implementation of the ACSI service

`ListMsg` node is a node that represents a list, with each member also a node. Our message tree is virtually a simplified version of a XML DOM tree. Using such kind of message representation, we can easily pass the internal events and service parameters/responses among different internal objects.

3.1.3 Service model

Figure 11 shows the internal architecture of the implementation of the ACSI services on the server side. Each active association is equipped with an instance of `AssocAPI`. `AssocAPI` provides the basic ACSI interface for an association instance. On receiving the application requests, the association instance invokes relevant services provided by `AssocAPI` to complete the tasks. `AssocAPI` is supported both by the underlying kernel services which provides the basic functions to complete the ACSI service requests, and by the communication utilities, which talks to J-Sim [11], a network simulator.

Instead of delivering the reports using multi-party application associations, we deliver them to the applications using the corresponding two-party application associations. The reporting procedure is rather straightforward: when the value of a data attribute is changed or updated, we check whether this operation satisfies the trigger condition. If the trigger condition is met, the data attribute issues an internal event to all the data sets marking it as a member and those data sets will forward the internal event to the active report control blocks, which generate the reports and send them to the subscribing applications. For the sake of simplification, we assume the members of a data set are all data attributes in our simulation.

3.2 Running the simulator

One can follow the following steps to create a simulation scenario and run a simulation:

Step 1 **create specialized classes based on the template classes**.
The simulation package provides the necessary template classes from `Server` to `DataAttr`. Users just need to derive a specialized class from

the appropriate template class and configure this class in the `configure` method as follows:

```
class DemoRelay
  extends LDevice {
...
protected void configure() {
declareLN(new DemoXCBR(``XCBR0''));
declareLN(new DemoMMXU(``MMXU0''));
}
}
```

Step 2 **create applications based on the `Application` class**.
Similar to the previous step, one needs to define his own specialized `Application` class:

```
public class DemoApp1
  extends Application {
protected void _start() {
...
}
}
```

Currently, our simulation package only supports talking to one server for each application instance. Enabling an application instance to talk to multiple servers remains our future work.

Step 3 **create the network topology in J-Sim**.
We shall build a network scenario to simulate. One could easily create the require network topology in J-Sim by following the INET Tutorial [1]. It should be noted that only TCP based communication is supported in current version of our simulator.

Step 4 **bind Application instances to the J-Sim nodes**.
After the above steps are done, we should bind the specialized Application instances to the J-Sim network nodes:

```
set server [java::new demo.DemoServer]
mkdir demo.DemoApp1 h3/app1
connect -c h3/tcp/up@
  -and h3/app1/down@

mkdir demo.DemoApp2 h4/app2
connect -c h4/tcp/up@
  -and h4/app2/down@
```

Step 5 **call `createTPAA` to create TPAAs and bind them to the J-Sim nodes**.
Then bind the TPAA instances to the J-Sim network nodes:

```
cp [$server createTPAA] h1/tpaa1
connect -c h1/tcp/up@
  -and h1/tpaa1/down@

cp [$server createTPAA] h2/tpaa2
connect -c h2/tcp/up@
  -and h2/tpaa2/down@
```

Step 6 **start J-Sim simulation**.
Finally, we can start the simulation of the IEC 61850 standard:

```
set sim [attach_simulator .]
# start the association instances
run h1-2
# then the application instances
run h3-4
```

4 Conclusions

Due to its complexity and the assumed domain-specific knowledge, the IEC 61850 standard is difficult for people other than domain experts to understand and implement. We presented our understanding of the standard as well as the design and implementation of our simulation of the IEC 61850 protocol based on J-Sim. Our experience and lessons of simulating the protocol show that although the IEC 61850 adopts an object-oriented approach, implementers still need their own internal data representation or take the advantage of a database system.

5 Acknowledgments

We would like to thank Zahid Anwar and Jianqing Zhang for their keen discussions on the IEC 61850 standard.

References

[1] J-sim inet tutorial, December 2003. `http://www.j-sim.org/drcl.inet/inet_tutorial.html`.

[2] BRAND, K.-P. The standard iec 61850 as prerequisite for intelligent applications in substations. *Power Engineering Society General Meeting, 2004. IEEE* (June 2004), 714–718 Vol.1.

[3] DNP USERS GROUP. The distributed network protocol. Website. `http://www.dnp.org/`.

[4] DOLEZILEK, D. Iec 61850: what you need to know about functionality and practical implementation. *Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, 2006. PS '06* (March 2006), 1–17.

[5] KIM, G.-S., AND LEE, H.-H. A study on iec 61850 based communication for intelligent electronic devices. *Science and Technology, 2005. KORUS 2005. Proceedings. The 9th Russian-Korean International Symposium on* (2005), 765–770.

[6] KOSTIC, T., AND FREI, C. Modelling and using IEC 61850-7-2 (ACSI) as an API. In *Power Tech 2007 Proceedings* (July 2007).

[7] KOSTIC, T., PREISS, O., AND FREI, C. Understanding and using the iec 61850: a case for meta-modelling. *Computer Standards & Interfaces 27*, 6 (June 2005), 679–695.

[8] MCDONALD, J. Substation automation. ied integration and availability of information. *Power and Energy Magazine, IEEE 1*, 2 (March and April 2003), 22–31.

[9] MODBUS-IDA. Modbus application protocol specification v1.1b, December 2006. `http://www.modbus-ida.org/`.

[10] SYSTEMS INTEGRATION SPECIALISTS COMPANY, I. IEC 61850 evaluation kit CD-ROM. Software. `http://www.nettedautomation.com/solutions/uca/evalkit/index.html`.

[11] TYAN, H. Y., AND HOU, C. J. Javasim: A component-based compositional network simulation environment. In *Proceedings of the Western Simulation Multiconference – Communication Networks And Distributed Systems Modeling And Simulation* (January 2001).

[12] UCA INTERNATIONAL USERS GROUP, I. Introduction to UCA®version 2.0. Tech. rep., Institute of Electrical and Electronics Engineerings, Inc., November 1999.

[13] UDREN, E., KUNSMAN, S., AND DOLEZILEK, D. Significant substation communication standardization developments. In *2nd Annual Western Power Delivery Automation Conference (WPDAC) Proceedings* (April 2000).