# Design of a Protection Relay Incorporating UCA2/MMS Communications

# DESIGN OF A PROTECTION RELAY INCORPORATING UCA2/MMS COMMUNICATIONS

**M Adamiak, D Baigent, R Moore, B Kasztenny, J Mazereeuw**
**GE Power Management, Canada**

## INTRODUCTION

The Utility Communication Architecture or UCA had its origins in 1990 as the framework for the ensemble of communication requirements that exist in the utility enterprise. It was at this time that utility managers were looking to consolidate communications among their planning, SCADA, metering, protection, and control departments. In attempting this consolidation, the cost of integration of diverse communication protocols was realized and a drive towards communication commonality was begun [1,2,3].

Since 1993, there has been a focus on the application of UCA in the substation. The process started with the creation of a requirements document that defined the communication requirements for the various functions inside a substation. The functional requirements document was followed by an implementation and evaluation phase that defined the "profile" of a communication structure for the substation. The profile that has been defined is shown generically in Figure 1. The goal in defining the "next generation" substation profile was to implement a high speed, networkable, peer to peer network using as many "existing" communication protocols as possible. In addition, the goal of user data interoperability required the definition of standard names for commonly used data objects.

The profile developed uses Ethernet for the Physical and Data Link layers. Although Ethernet is "non-deterministic" when operated in a "shared access" mode of operation (due to collisions), Ethernet technology has advanced to provide "switched" access which minimizes collisions. In addition, Ethernet provides a growth path to higher-speed Ethernet networks such as 100MB and 1GB with 10GB already defined.

For the "networking" layers, although the original goal was to stay within the realms of the International Standards Organization (ISO) standards, the popularity of the Internet dictated the inclusion of the TCP/IP networking layers. In November of 1999, the International Electrotechnical Committee (IEC) selected TCP/IP as the "mandatory" networking protocol for intra and inter substation communications and the ISO networking layers as optional. The inclusion of these networking layers makes data from the substation available over a utility intranet, WAN, or even the Internet.

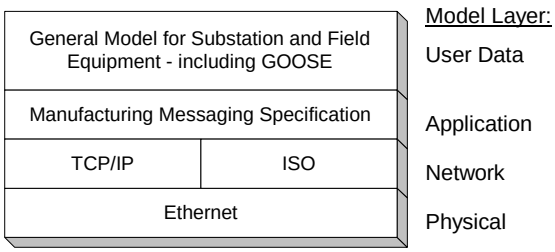


Figure 1: UCA substation profile.

For the Application or service layer, the Manufacturing Messaging Specification (MMS) was chosen. MMS provides a rich set of services to read, write, define, and create data objects. It is MMS and its ability to manipulate logical objects that differentiates this profile from all other existing profiles.

Lastly, the UCA substation profile defines standard "object models" for commonly used data elements. These standard models are defined in the document entitled: General Object Models for Substation and Field Equipment (GOMSFE). This standardization facilitates interoperability as any manufacturer who allows "Phase AG Voltage" to be externally visible, does so in a common manner.

This paper now details the hardware and software considerations, issues encountered, and benefits enjoyed by the authors and others in implementing the substation profile of UCA in a digital device.

## HARDWARE CONSIDERATIONS

The primary functional requirements that had hardware implications were scalability, reliability, and performance. These items became guideposts in the design of next generation protection platform - the Universal Relay (UR [4]). The scalability need was recognized in terms of communication channel's bandwidth as well as the protection processors ability to process data and advanced MMS functions. For example, at the time of the relay design, "hardened" Ethernet was only available at 10MB. Today, next generation hardened Ethernet controllers are available that operate at 100MB. In addition, it was recognized that some of the advanced MMS services such as "create object" would require additional memory and processing capability. To address the ability to make these changes, the entire UR design was done on a modular basis. As such, when a new CPU is available, when more memory is needed, or when the 100MB Ethernet communication controller is available in the extended temperature range, the user only has to

change out the CPU module to obtain the required performance enhancement (Figure 2).

The ability to perform relay to relay communications with the UCA substation profile has enabled the implementation of many innovative protection and control applications that use messaging over the wire. Many, if not all, of these applications are of a critical nature. The criticality of communications has raised the issue of reliability or communications within the substation.

To address this issue, a redundant Ethernet communication architecture has been utilized [4]. The design incorporates a dynamic "media switch" that can automatically detect a "loss of link" on the primary Ethernet channel and within milliseconds, automatically switch to the back-up Ethernet channel.

Lastly, the choice of the Ethernet controller hardware can affect performance of message handling in the relay. In particular, the ability to perform hardware filtering of the link addresses can off-load the CPU from communication processing and correspondingly, allow the processor to prioritize messages when received. Speed of message handling is important in order to meet the UCA target of 4ms for messaging processing of a digital message. A choice of CPU hardware was made to meet these criteria [4,5].

## SOFTWARE CONSIDERATIONS

As mentioned previously, one of the goals of in the creation of the UCA substation profile was to use as many "off the shelf" standards as possible. The effect of achieving this goal is that most of the software needed to perform the networking and application layer tasks is available off the shelf from multiple vendors. The presented solution availed itself of this fact as was able to purchase "off the shelf" MMS and ISO networking software from SISCO [6].

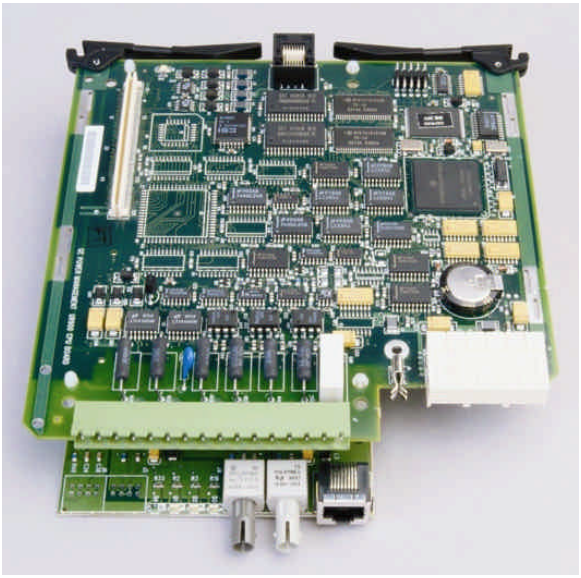

Figure 2: Modular CPU with fiber Ethernet Port(s).

The above software components basically load onto the relay platform. As our platform is based on a real time operating kernel, the internals of the operating system take care of task scheduling and linking with other tasks. Code space for these programs runs about 128kbytes. In addition to the code space, dynamic memory has to be allocated. When a request for data is received, the "response" is generated in this dynamic memory area. As the relay can serve multiple clients, sufficient memory needs to be allocated to accommodate the requirements of each client.

The challenge in the implementation comes in the mapping of the internal relay data objects into the standard GOMSFE objects. To facilitate this mapping, the SISCO software contains a program called the "Object Foundry". This program allows the programmer to create the object structure and tree of the GOMSFE objects that will be made visible to the outside world. See Figure 3 for an example mapping. Once the outside object list is defined, it is a matter of simple mapping of the relay internal variables / object models to the GOMSFE object model. Adherence to the GOMSFE object models insures the interoperability of data among the various manufacturer's devices.

## OTHER CONSIDERATIONS

Although the goal in choosing the various profile layers for the substation implementation of UCA was to use "existing protocols", some enhancements were needed to meet the required functionality. One of these functions, in particular, was high-speed device to multi-device communications. The MMS information report service was used to achieve this functionality and it was used to deliver a binary object model known as the Generic Object Oriented Substation Event or GOOSE.

The GOOSE message was implemented as a connectionless ISO datagram. As a connectionless message, the Media Access Control (MAC) address of the sending device is known and subsequently loaded into the header of the GOOSE message. Also included in the header is the name of the sending device, time of the event that launched the GOOSE, and the expected time of the next GOOSE message. Data states are sent in pairs with (0,1) and (1,0) being the primary logical states, (0,0) being defined as a "transition" state, and (1,1) being unde-

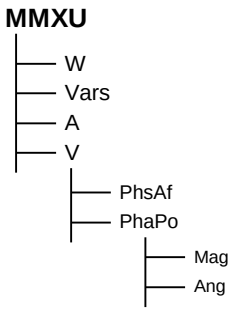


Figure 3: Object model structure from object foundry.

fined.

Each node receiving a multi-cast message needs to be able to determine from whom the message came and whether the data received was of interest to the receiving device. In establishing the procedure for how this would happen, the authors worked with SISCO to develop the **S**elf **M**entoring **A**nd **R**e-**T**raining or SMART GOOSE.

Self Mentoring is the process of finding out the address of the GOOSE sender to which a device would like to listen. During set-up, the engineer programs the receiving device with a list of devices from which it should expect to receive data. On start-up of the network, the Ethernet receiver in the device goes into "promiscuous" mode whereby all multi-cast messages are read and decoded. The name of the device sending the message is compared with the programmed list of "devices to listen to". If the names match, the receiving device stores the MAC address of the sending device in a high-speed hardware address comparator. Once all address / name matches have been made, promiscuous mode is turned off and all multi-cast messages are now captured based on a hardware address comparison.

The "Re-Training" part of the SMART GOOSE comes into play when a relay is taken out of service or a CPU module is exchanged or upgraded. When the CPU card is changed, the corresponding MAC address for the Ethernet card changes as each Ethernet controller in the world has a unique 48-bit address. SMART GOOSE (on the receiving side) recognizes that the GOOSE message it was expecting is missing. In this scenario, the receiving relay goes back into promiscuous mode – again searching for a message with the name of a desired device inside. Once found again, the new MAC address for the new CPU / Ethernet controller is stored in the high-speed look-up table and the receiving relay once again turns off promiscuous mode and returns to normal operation.

## CHALLENGES FACED

As is to be expected with any new development, there were challenges that had to be addressed and performance issues to improve. A few of these challenges are highlighted here.

### GOOSE Performance

One particular performance challenge was with GOOSE messaging. In the original implementation, GOOSE messages were processed in the same thread with all other MMS data requests. As such, GOOSE message delivery times were averaging 15ms – somewhat beyond the goal of 4ms established by the UCA specification team.

In revisiting the GOOSE implementation, a refinement was made to allow the GOOSE decoding software to operate as a separate thread in the processor. Making this change allowed received GOOSE messages to be processed separately from all other MMS messages and consequently, allowed GOOSE messages to be processed at a higher processing priority in the CPU. Additionally, the encoding and decoding of GOOSE messages was optimized savings additional milliseconds of GOOSE processing time. As a result, the presented solution was able to increase the speed of their GOOSE message 3 fold. Complete digital input to digital output timing includes digital de-bounce time on the input and contact actuation time on the output. Figure 4 shows an actual relay input to relay output timing diagram.

### TCP/IP Connection Time-Out

As mentioned earlier, the networking layers of TCP/IP have become the primary choice for the IEC and as such, will see much more use in the substation. Keeping in mind that TCP/IP was designed for the Internet, it bring with it an undesirable feature – its propensity to hang on to its client connections. If a client crashes in a substation, even though the MMS server software in the relay in not receiving any data requests, the TCP "keep alive" timer does not relinquish the connection for up to 2 hours. Even though multiple connections can be established with a server, its maximum is quickly reached.

Once identified, the solution to this problem came in the form of maintaining independent timers on data requests from a client. If data requests were not detected for over 2 minutes (our arbitrary setting), communications is automatically shut down and restarted thereby allowing the client to re-connect when ready.

### Things that go Bump in the Night…

The MMS software was written in "C" language and tested on a particular operating system. Often when one ports a 3rd party library onto a new operating system, there are numerous configuration issues. Fortunately for the authors, SISCO had already ported the MMS services to the operating system of the UR and *most* of the configuration issues had been resolved.

As mentioned earlier, MMS does dynamic memory allocation to meet its clients requests. It is very important that the memory allocated for the request be de-allocated

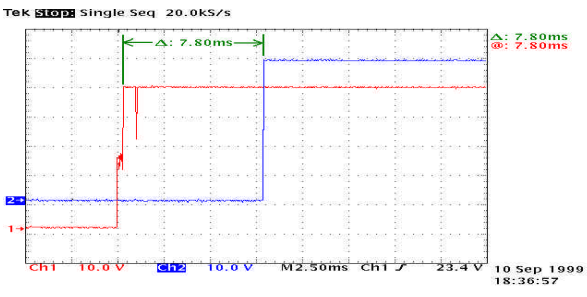


Figure 4: Digital input to digital output GOOSE timing.

after the request is fulfilled. Clearly, one can quickly run out of memory – especially on invalid requests.

Data modeling became a dynamic aspect of MMS implementation. As with any great work of art, there is constant refinement to get it just right. Unfortunately, the software must follow suit. Luckily, the primary data models have not changed much and as a result, any applications built using the original mapping remain basically unaffected. Undoubtedly, the data models will continue to change, however, the beauty of the object model concept is that is enables model extension without necessarily changing the original model.

Limit testing was always an adventure into the unknown. For example, if the code was limited to a read of 100 variables, what does it do if you ask for 101 variables? Needless to say, such tests helped to harden the basic software. SISCO was very responsive to all our requests and a close working relationship has subsequently been developed.

## BENEFITS AND NEXT STEPS

The clear winner overall is going to be the user. In initial contacts with system integrators, the ability of the system to "self define" its data objects and the commonality of objects among manufacturers has been a tremendous time saver. Since all the data items are logically defined, the addition of a new variable does not change the memory map of the device. Contrasting this to Modbus where the addition of a variable must be defined in the user application and may cause other items to shift. In addition, there is no universal mapping of data elements to Modbus registers among different manufacturers. As such, the integration of each Modbus device is unique (and subsequently, an integration task and an integration cost).

Another major customer benefit is the ability to perform virtual training and debugging over the Internet, or if access is made available, remotely over the customer's intranet. This capability is made possible through the ability of UCA to be networked and the ability of the relay to support multiple clients. A manufacturer's application engineer can be on-line simultaneously with the customer and guide him/her through a setting procedure or examination of settings, events, or even oscillography.

It is clear that as time rolls on, additional UCA features will be requested / required and performance upgrades will be required. For starters, 100MB Ethernet will shortly be migrating into the substation. Chips that can support 100MB Ethernet over the extended temperature required for substation operation are starting to appear.

As integrators begin using GOOSE to implement substation control schemes, undoubtedly additional features and functions will be required.

Although one object models will contain many variables, data for a Human Machine Interface (HMI) screen often requires pulling data from multiple object models. It has been pointed out that if all the required data was in one object model (such as a User Defined model), the communication bandwidth required would be greatly reduced.

The final mark of success will be the world-wide adoption of UCA as the profile of choice for next generation substation automation. Major inroads to achieving this goal have been made through the introduction of the UCA concepts into the IEC. Technical Committee 57 of the IEC is in the process of creating a standard (IEC 61850) that has its foundation in the work done for UCA. Once approved, the concept of networked communication based on MMS and Object Models will become an international standard.

## SUMMARY

The definition of UCA was done in a top-down manner and, as such, resulted in a very well thought out profile. Applying these same principles in the design of the target device can result in optimized performance today and growth potential for tomorrow. Overall, implementation of UCA has been relatively "bumpless". The benefits of UCA have been observed by many and will aid in the acceleration of acceptance of the profile.

## REFERENCES


1. Adamiak M, Premerlani W, "The role of utility communications in a deregulated environment", 1999, International Conference on System Sciences, Hawaii, January 5-8.
2. "Substation Integrated Protection, Control, and Data Acquisition. Phase 1, Task 2: Requirements and Specification", EPRI – RP3599-01.
3. "General Object Model for Substation and Field Equipment (GOMSFE)", ftp://sisconet.com/epri/UCA2.0/GOMSFE9.zip
4. Pozzuoli M P, "Meeting the Challenges of the New Millennium: The Universal Relay", 1999, Texas A&M University Conference for Protective Relay Engineers, College Station, Texas, April 5-8.
5. Adamiak M, Baigent D, Evans S, "Practical considerations in application of UCA GOOSE", 2000, Georgia Tech Relay Conference, Atlanta, USA, May 3-5.
6. "Common Application Service Models (CASM) and Mapping to MMS", ftp://sisconet.com/epri/UCA2.0/CASM_15.ZIP.