Application of Ethernet Fieldbus to Substation RTU and Automation Networks

David Dolezilek, Francisco Chumbiauca, and Michael Rourke
Schweitzer Engineering Laboratories, Inc.

Presented at the
13th Annual Western Power Delivery Automation Conference
Spokane, Washington
March 29–31, 2011

Application of Ethernet Fieldbus to Substation RTU and Automation Networks

David Dolezilek, Francisco Chumbiauca, and Michael Rourke, *Schweitzer Engineering Laboratories, Inc.*

Abstract—**Unlike remote terminal unit (RTU) installations in the past, which were largely used to report I/O values and transmit supervisory control and data acquisition (SCADA) controls, present day substation automation networks additionally coordinate high-speed load management, intelligent electronic device (IED) integration, and user security. Substation operators are faced with many technology choices in an effort to balance ease of integration and system performance.**

This paper discusses the application of an Ethernet-based fieldbus technology, initially developed for industrial control systems, to substation automation networks. EtherCAT is an open fieldbus standard that combines the flexibility of Ethernet connectivity with real-time deterministic performance and a highly efficient messaging structure. In the substation environment, EtherCAT can be used in a variety of network topologies to improve the performance and reliability of the power system.

This paper provides a brief tutorial on the functionality of EtherCAT messaging and communications network topologies. Additionally, EtherCAT data acquisition and control performance are compared to other contemporary Ethernet and serial messaging technologies in typical electric utility substation applications.

I. Introduction

To accommodate new, increasingly popular intelligent electronic device (IED) network functions, substation communications infrastructure is experiencing a dramatic change and is migrating to Ethernet. The majority of successful substation integration systems that are going into service today and in the near future are based on non-Ethernet local-area networks (LANs), built using EIA-232 point-to-point and EIA-485 multidrop communications ports within the IEDs. The information exchanges are carried out using register- and/or address-based protocols, such as DNP3, IEC 60870, and Modbus®. These communications methods also include National Institute of Standards and Technology-approved protocol standards created by a standards-related organization (SRO) and offered via a "reasonable and nondiscriminatory" license. This includes MIRRORED BITS® communications, additional open vendor-developed serial protocols, and other standards, such as IEEE C37.94. With the new IEC 61850 standard and the popularity of Ethernet networks, the entire picture of substation communication is changing.

In this paper, we illustrate the similarities and differences of MIRRORED BITS communications and IEC 61850 Generic Object-Oriented Substation Event (GOOSE) messages in order to explain the functionality of the latter. Then we compare and contrast shared Ethernet bandwidth techniques using IEC 61850 GOOSE with dedicated nonshared Ethernet bandwidth techniques using EtherCAT to show the improvements of time-deterministic communications.

II. Contemporary Substation Ethernet Relies on IEEE 802.1 and ISO/IEC 15802-1

Prior to switched Ethernet technology, early networks were built using Ethernet hub networks based on IEEE 802.3 Carrier Sense Multiple Access/Collision Detection (CSMA/CD), where messages competed for bandwidth in collision domains [1]. With the use of twisted pairs and fiber cables that separate the transmitted and received traffic, modern switched Ethernet LANs create a truly full-duplex and collision-free communications environment. IEC 61850 migrated to IEEE 802.1 and ISO/IEC 15802-1 in order to change from the network behavior associated with IEEE 802.3 CSMA/CD and collision domain network segments. ISO/IEC 15802-1 defines the Media Access Control (MAC) Service used in modern Ethernet navigation [2]. The MAC Service provides transparent transfer of data between MAC Service users by directing messages from one port to another on the network until the message reaches its final destination. The 48-bit hardware MAC (hMAC) address is divided into two parts. The first 24 bits correspond to the organizationally unique identifier (OUI), as assigned by the IEEE Standards Board. The second 24 bits of the address are administered locally by the assignee to provide uniqueness.

After an initial and nondeterministic network configuration process, Ethernet switches learn and archive a list of the MAC addresses of each device to which they direct traffic. When receiving a message from a port, the switch examines the destination MAC address of the message and forwards it only to the port configured to redirect messages that match the MAC address. This method works for client/server Internet Protocol (IP) traffic, such as supervisory control and data acquisition (SCADA) poll and response using Manufacturing Message Specification (MMS), Modbus TCP/IP, or DNP3 LAN/WAN; however, it does not work for IEC 61850 GOOSE.

The original MIRRORED BITS communications messages exchanged Boolean information over physically segregated point-to-point communications channels. In 2000, Utility Communications Architecture (UCA2) members saw the value of this peer-to-peer virtual wiring method and created multicast Boolean exchange over Ethernet, called UCA2 GOOSE. IEDs were capable of reliable generation and consumption of UCA2 GOOSE messages; however, the inefficiency of navigating IEEE 802.3 CSMA/CD collision domain Ethernet networks made the use of UCA2 GOOSE unreliable.

Shortly after the MIRRORED BITS communications message was modified to convey Boolean, analog, and engineering access textual information, UCA2 GOOSE was similarly modified. During this time, UCA2 was merged into the IEC 61850 communications standard, and UCA2 GOOSE was renamed IEC 61850 General Substation State Event (GSSE). A new message was designed to transfer both Boolean and analog data types, but not engineering access text. This new message, IEC 61850 GOOSE, was also changed to use the ISO/IEC 15802-1 MAC Service. As such, IEC 61850 behaves in a multicast mode without knowledge of the destination hMAC addresses and uses multicast MAC, or virtual MAC (vMAC), instead. Therefore, IEC 61850 GOOSE messages are published to a group destination multicast vMAC address, which goes to every port.

An Ethernet switch processes every message received or transmitted by each port. It takes time for switches to process messages, and this introduces a short, but unavoidable, switch processing latency delay. If a switch cannot process and forward all of the messages that it receives, a backlog occurs. A message waits in a transmitting memory queue for its turn to be sent. If this occurs, there is a switch queue latency in addition to the switch processing latency. A message may need to go through several switches in a network to reach its destination. When networks are designed with knowledge and care, the likelihood of a switch queue delay is minimized, but not eliminated.

An Ethernet switch stores and forwards messages as they are received over Ethernet cables, similar to the behavior of a MIRRORED BITS communications switch using serial cables. Layer 2 GOOSE messages have a multicast address, not a destination address, and therefore cannot be managed via mechanisms for MMS, DNP3 LAN/WAN, Telnet, File Transfer Protocol (FTP), and other IP messages using Layer 3 and above. Multicast (one-to-many) means that each time a GOOSE message is received on a port, it is automatically sent to every other port. Even though GOOSE no longer requires collision detection and mitigation among Ethernet messages within a collision domain, the use of shared Ethernet bandwidth provisioning prohibits deterministic, synchronous GOOSE message exchange.

III. Ethernet Traffic Navigation Techniques to Compensate for Shared Bandwidth Behavior

One of the techniques to alleviate the network burden of multicast/broadcast messages is the virtual local-area network (VLAN). IEEE extended the Ethernet Standard 802.1 with the designator Q for message quality, which includes extensions for optional VLANs via a previously unused field in the Ethernet header tag that becomes a VLAN identifier (VID). IEEE 802.1Q VLAN, or QVLAN, divides a physically connected network into several VLANs. While keeping the sensitive information private, QVLAN techniques can restrict traffic flow of multicast and/or broadcast messages to a single QVLAN and therefore the devices within it.

IEC 61850 adopted the use of the IEEE 802.1Q VID as a QVLAN tag to identify multicast messages and overcome the inability to perform network routing by performing manual routing. Because of the unwanted and unstoppable automatic distribution of multicast messages, the manual routing acts in reverse. The multicast messages are routed everywhere but are only allowed to pass through ports from which they have not been blocked. In IEC 61850 networks, QVLAN tags are implemented within the multicast message by the publishing IED and used by switches for manual routing. This is one of several network processing tasks that have been forced into the IEDs to compensate for inadequate data flow capabilities in Ethernet networks. Switches unable to perform QVLAN filtering, or those configured incorrectly, will not work properly and may block even wanted GOOSE transfer. Best engineering practice methods within IEC 61850 dictate a unique QVLAN identifier for each GOOSE message publication.

The only effective method to segregate Ethernet multicast traffic and make GOOSE message navigation behave as deterministically as possible is to follow these simple rules:

- Assign each GOOSE message a unique QVLAN.
- Assign each GOOSE message a unique vMAC.
- Assign each GOOSE message a unique application identifier (app ID).
- Assign the last octet of the vMAC, app ID, and QVLAN the same value.
- Allow no multicast messages on the network without QVLAN tags.
- Disable all unused switch ports.
- Configure each switch port to use MAC filtering to block delivery of every multicast message to the connected IED, except the GOOSE that the IED has subscribed to within its Substation Configuration Language (SCL) file.
- Configure each switch port to use QVLAN filtering to block delivery of every multicast message to the connected IED, except the GOOSE that the IED has subscribed to within its SCL file.

IV. EtherCAT Technology Eliminates Shared Bandwidth Behavior

A. *EtherCAT Overview and Technology Introduction*

1) *Level 0 Fieldbus in Process Control*

Concurrently with the development of Ethernet networks for substations, process control engineers have been considering Ethernet applications for industrial systems. Ever since the introduction of the programmable logic controller (PLC), users have looked for methods to more efficiently connect field I/O with controllers and construct larger distributed control networks. Many of the techniques used have been similar to substation networks.

2) *S95 or Purdue Network Model*

The terminology commonly used to describe components of industrial networks differs from terms we use for substation networks. A few of these terms are helpful as part of this discussion. The S95 (or Purdue) model, as seen in Fig. 1, represents the hierarchy and nomenclature of networking systems within industrial manufacturing organizations [3].

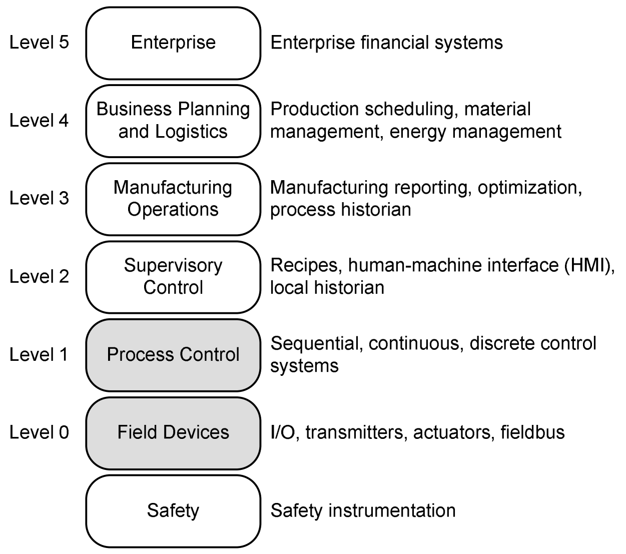

Level 5
Enterprise
Enterprise financial systems
Level 4
Business Planning and Logistics
Production scheduling, material management, energy management
Level 3
Manufacturing Operations
Manufacturing reporting, optimization, process historian
Level 2
Supervisory Control
Recipes, human-machine interface (HMI), local historian
Level 1
Process Control
Sequential, continuous, discrete control systems
Level 0
Field Devices
I/O, transmitters, actuators, fieldbus
Safety
Safety instrumentation

Fig. 1. S95 network hierarchy model

Level 0 and Level 1 networks are similar in many ways to the process bus and station bus concepts that we apply to substation networks, as shown in Fig. 2. Under the S95 model, current transformers (CTs), potential transformers (PTs), and temperature sensors are all considered Level 0 devices. Remote terminal units (RTUs) and protective relays are categorized as Level 1.

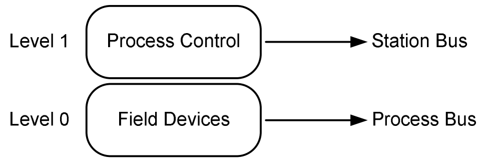


Fig. 2. Relative comparison of S95 model to substation networks

3) *Evolution of Level 0 Networks*

More than thirty years ago, process control systems began transitioning from hard-wired relay logic and single-function loop controllers to PLC-based Level 1 systems [4]. The limited computing capability of early PLCs supported a relatively small number of field inputs and outputs. Because of that limitation, I/O wiring could be economically terminated within the same cabinet as the PLC. However, as automation vendors developed more advanced PLCs that accommodated many more field points—and a larger variety of signal types—users needed an alternative I/O networking method in order to avoid terminating hundreds or thousands of points directly in the controller cabinet.

In response, equipment vendors developed I/O devices that could communicate with a PLC via a specialized network, or fieldbus. Fieldbus networks were based on EIA-232 or EIA-485 communications and provided a means for a controller to read and write large quantities of I/O points via a single communications cable. Early fieldbus implementations were based on proprietary network protocols; but in the 1990s, a number of standard protocols, such as FOUNDATION fieldbus, became popular in the industry.

B. *Development of EtherCAT*

The serial networking used for fieldbuses in the 1990s became bandwidth limited as controllers with larger and larger processing capabilities were introduced into the market [5]. Use of Ethernet technology as a second generation fieldbus was a natural evolution, but the inherently nondeterministic nature of Ethernet created difficult new problems. Additionally, most Ethernet equipment was not designed for industrial use, which made the adoption of Ethernet-based fieldbuses even slower.

A number of Ethernet protocol standards exist that operate under constraints intended to make them more deterministic. Some standards use messaging rules to eliminate collisions and reduce the signal jitter. Others take advantage of Ethernet switches that use VLANs in order to reduce network burden within a subnet to only messages sent from or meant for devices within that subnet.

While these methods improve network determinism, they still do not make efficient use of the bandwidth. Most of these protocols adhere to an Ethernet paradigm that each device sends an entire Ethernet frame for each message and every message delivered to a device is composed of an entire Ethernet frame. Each frame can be as long as 1,518 bytes, even if the usable process information (either inputs or control outputs) only requires a few bytes. The result, even when using multicast messages, is that a large amount of the network traffic is consumed by administrative information.

C. *EtherCAT Technology*

The developers of EtherCAT created solutions for both the time and efficiency challenges of Ethernet [6]. The fundamental difference between EtherCAT and other Ethernet fieldbus protocols is that a single EtherCAT frame contains I/O point updates from many devices in a network, not just a single device. Existing transport protocols could not accomplish this, so a new EtherType was defined explicitly for the EtherCAT protocol. As we will see, this approach provides complete compatibility with Ethernet standards.

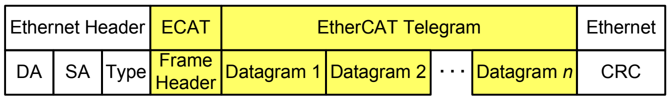
1) *Development Objectives and Requirements*

Even though it is an open standard protocol today, EtherCAT began as the invention of Beckhoff Automation. The main development objectives were as follows:

- Deterministic network operation.
- Fast network update time.
- Efficient use of network bandwidth.
- Compatibility with existing controller Ethernet hardware.
- Full conformity with the Ethernet standard.
- Economical implementation for small and large I/O devices.

2) *Definition of EtherType for EtherCAT Messages*

The EtherCAT frame (shown in Fig. 3) is specifically designed to incorporate process data from many Ethernet nodes into a single message. The telegram can extend to multiple frames, with a maximum size of 4 gigabytes. Individual devices in the network are configured to read and write data from specific regions of the telegram, which means that the telegram mapping sequence is not directly related to the physical network configuration.



ECAT – EtherCAT
DA – Destination Address
SA – Source Address
CRC – Cyclic Redundancy Check

Fig. 3. Standard Ethernet frame for EtherCAT messages

As shown in Fig. 4, the independent data mapping allows designers to create telegrams based on specific process sequences or mapping preferences in the PLC or other Level 1 controllers.

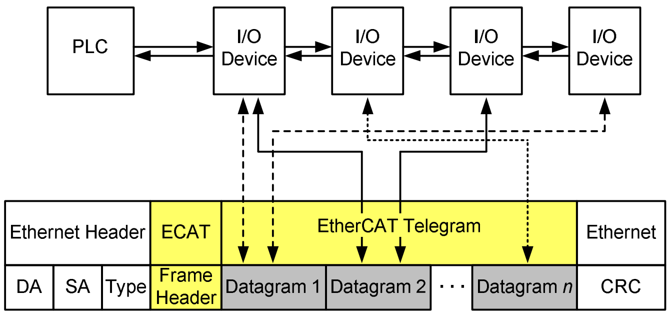


Fig. 4. Network location independent from EtherCAT mapping

3) *How On-the-Fly Processing Works*

In order to achieve the needed network speed for critical applications, EtherCAT devices use a low-level, on-the-fly processing method where all devices within a network segment receive the entire EtherCAT message. The PLC, or EtherCAT master, begins the sequence by sending a message with updated output control data and input status from the previous data cycle.

As the first I/O device in the network starts receiving the frame, it automatically reads the proper control data from the telegram and writes updated process data into the telegram. The first device also automatically forwards the updated telegram to the next I/O device with a delay of less than 1 microsecond. Each subsequent device similarly reads and writes the needed portions of the telegram; then the last device returns the completed message to the PLC. Even in very large systems, the entire round trip can be completed in less than 100 microseconds.

The hardware interface for each device consists of standard Ethernet ports. However, each I/O device has a field-programmable gate array (FPGA) or low-cost application-specific integrated circuit (ASIC) that reads and writes the EtherCAT telegram such that the incredibly low signal delay is maintained. A fieldbus memory management unit (FMMU) process in the FPGA or ASIC is automatically configured upon network initialization with the location of relevant input and output locations in the EtherCAT telegram. Once the network enters normal operating mode and starts transmitting telegrams, no time is wasted evaluating the entire telegram. The FPGA or ASIC can quickly read and write just the needed memory locations and forward the telegram. After sending the telegram, the device internally acts on newly received control commands and updates inputs before the next telegram arrives.

The PLC maintains overall control system determinism by starting each telegram transmission on a fixed schedule. Low processor jitter in modern control hardware enables a repeatable schedule.

4) *EtherCAT Works Over Switched Networks*

EtherCAT exhibits maximum performance on a dedicated network segment, as shown in the previous section, but the protocol specification also supports use on a multipurpose Ethernet network using User Datagram Protocol/Internet Protocol (UDP/IP) messages. As shown in Fig. 5, the EtherCAT telegram is simply encapsulated for the UDP application. While this use case illustrates the flexibility of EtherCAT, users need to understand that the EtherCAT update rate degrades over a switched network. The amount of degradation depends on the Ethernet switches in service and the overall network performance.

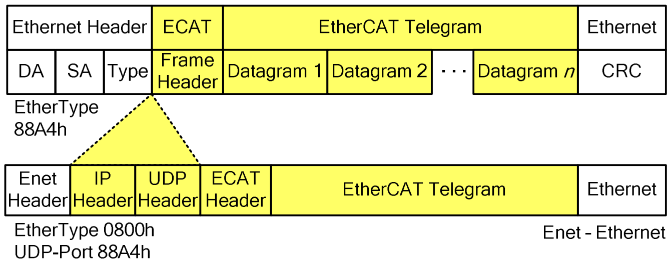


Fig. 5. EtherCAT messaging via a switched network

5) *How Multiple Vendor Products Work on Network – Open Protocol*

The combination of a strong set of conformance tests and the standard hardware interface (via FPGA or ASIC) results not only in interoperable devices between many vendors but also in consistent device performance in an EtherCAT network. A Level 1 control device can be installed with digital I/O modules from one vendor, analog I/O from a second vendor, and motor drives from a third. As long as each device is EtherCAT compliant, the network operates properly.

V. Performance Testing

Two tests were conducted in order to compare the round trip performance of a standard switched Ethernet network and an EtherCAT network. Each test consisted of the following two parts:

- Part 1 of the test was performed between two devices connected to the network to measure the time it would take to exchange I/O information.
- In Part 2, additional devices were added to the network to determine how the communications between the two initial devices would be affected.

A. Test 1 – Switched Ethernet Network

Two devices, A and B, were connected to an Ethernet switch, as shown in Fig. 6. Device B was constantly monitoring the status of a digital input on Device A, using IEC 61850 GOOSE messages. When the digital input asserted, Device B was notified and immediately sent a GOOSE message back to Device A in order to activate a digital output.

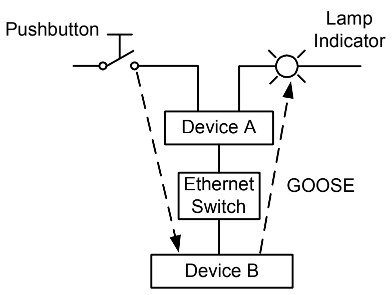


Fig. 6. First test configuration

The propagation time of the information from the assertion of the input to the activation of the output was measured using the 1-millisecond resolution Sequential Events Recorder (SER) in Device A. The time was calculated by subtracting the time stamp of the input from the time stamp of the assertion of the output. The input was generated several times in order to create multiple events. For each one of the events, the propagation time was 10 milliseconds.

Then the network was modified. Four additional devices were connected to the switch, each of them exchanging input and output information with Device B. This means that there were five devices attempting to communicate with one single node (Device B), as can be seen in Fig. 7.

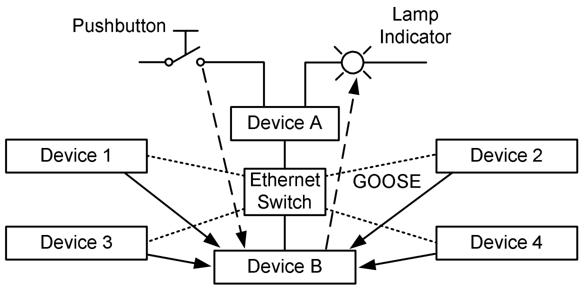


Fig. 7. Additional devices added to the network to increase traffic to Device B

Under these new conditions, the experiment was repeated. The digital input on Device A was activated several times to create multiple events in order to measure the message propagation time from Device A to Device B and back to Device A. This time, the result obtained was different from the first experiment. The values obtained from each event were different. In twenty activations of the input, the minimum propagation time registered was 10 milliseconds (same time as in the first experiment), the maximum propagation time was 16 milliseconds, and the mean was 12 milliseconds.

As we can see in this example, the response time changes when the network load is increased. The variability is due to each device on the network sending an independent Ethernet frame to Device B.

B. Test 2 – EtherCAT Network

Next, we tested an EtherCAT network in a similar way. An IED (Device A) read the status of a digital input from a remote module (Device B) and activated a remote digital output when the input asserted. The architecture is shown in Fig. 8.

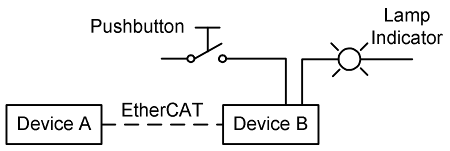


Fig. 8. Device A reads input and writes output on Device B

The propagation time of the EtherCAT frame was calculated using the time stamp of a 1-millisecond SER. As in the previous test, the digital input was activated several times in order to register multiple events. For this experiment, the minimum propagation time registered was 9.3 milliseconds and the maximum was 12.3 milliseconds.

Then four more devices were added to the EtherCAT network to increase the amount of data sent to Device A. Each additional device sent digital input statuses to Device A, as shown in Fig. 9.

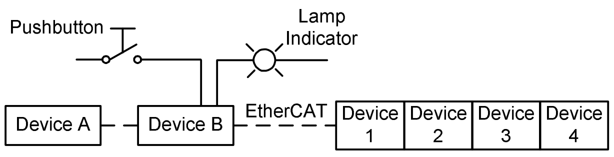


Fig. 9. Four additional devices added to the EtherCAT network to increase information traffic to Device A

Multiple events were triggered and logged in the SER. The results were identical to the original test system. The minimum time was 9.3 milliseconds, and the maximum was 12.3 milliseconds. In other words, the additional signals in the network did not affect the propagation time of the data.

Some of the reasons that explain this result are as follows:

- There is a single master on the network instead of multiple devices attempting to communicate independently with one single device.
- Slave devices connected to the network do not read and process the complete Ethernet frame. Each slave extracts and inserts only the relevant data while the frame passes through it.
- EtherCAT frame processing is done independently of the response time of any microcontroller.

VI. Processing of Input Messages

Frequently, in RTU applications, many digital inputs need to be processed to evaluate the state of substation equipment. As an evaluation of such an application, we compared the processing of GOOSE messages and EtherCAT telegrams in an RTU central processing unit (CPU). For each protocol, the CPU received 2,592 digital inputs.

For the case of using GOOSE as the I/O network, if there are 48 I/O modules that each report 54 digital inputs, the CPU can subscribe to and receive all 48 GOOSE messages every 10 milliseconds. Because each GOOSE message contains 1,518 bytes, the CPU needs to evaluate 72,864 bytes of data.

Using the same CPU to similarly receive the inputs via EtherCAT, we use 108 digital input modules. All of the modules reside on the same network segment, and the CPU can receive all inputs every 500 microseconds. Because all of the modules share EtherCAT messages, reducing overhead, the CPU evaluates only 4,500 bytes of data.

VII. Benefits of EtherCAT Networks

High-speed, distributed RTU, control, and protection schemes depend on deterministic network operation and fast processing. For example, utility implementations of distributed fast bus trip algorithms have been infrequent due to the expense and complexity of the equipment. EtherCAT networks provide a means to deploy an economical bus supervision and fast bus trip system. As shown in Fig. 10, the EtherCAT network monitors each feeder and disconnect switch for a sample substation bus. This topology allows the controller to quickly identify a fault that is cleared by a feeder relay or incoming transformer protection. The bus protection uses this information to delay tripping of the bus unless no monitoring unit identifies the fault. The bus relay can clear the entire bus quickly in case of an internal fault or breaker failure on a feeder.

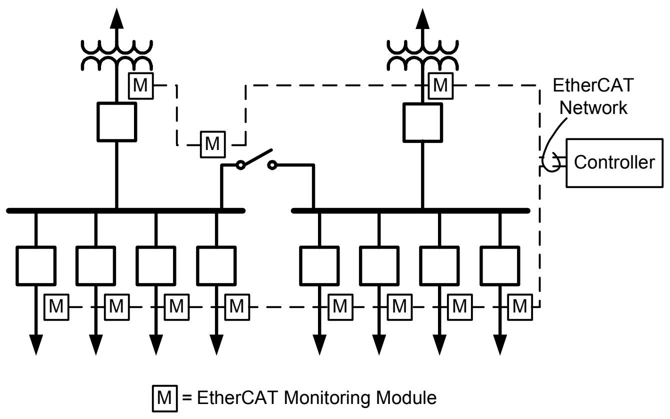


Fig. 10. Bus network overview

A. Improved Network Availability

EtherCAT natively includes diagnostics that can help improve control system operational security and availability. Each I/O module includes a watchdog triggered by valid EtherCAT messages. If the watchdog expires, the module declares a network error. I/O modules exist that use this error case to deassert all outputs in order to avoid unintended changes of state. Some vendors have I/O modules that include a further supervisory watchdog that monitors the FPGA or EtherCAT ASIC. The supervisory watchdog deasserts outputs in case of a module processing error.

The network controller also monitors network-wide system health in the following two ways:

- Each EtherCAT message includes a working counter value updated by each module in the network. If this counter returns to the controller with something other than the expected value, the controller immediately knows there is a network error. Users may determine how their system should respond to these network errors.
- Each module can report status information as part of an EtherCAT message. The status includes tags concerning the health of the module hardware and the network. The combination of the working counter and module status allows users to quickly identify and diagnose I/O system issues.

B. No Performance Degradation for Multivendor Network

The design of the EtherCAT message structure and network interface provides built-in interoperability when users install I/O modules from multiple vendors. Whether a given module has a very simple function and little data processing or includes complex algorithms via an independent processor, the network interface for EtherCAT is standard. Once a user configures the network membership and the network traffic commences, the I/O update time is consistent and measurable. Because every tag is updated during each message, the overall update rate does not vary depending on the dynamics of the system under control.

VIII. Conclusion

In this paper, we have shown how EtherCAT can be effectively used for real-time RTU and remote I/O applications. The design of EtherCAT provides the flexibility of Ethernet and operates deterministically.

IX. References

[1] D. Dolezilek, "Using Information From Relays to Improve the Power System – Revisited," proceedings of the Protection, Automation and Control World Conference, Dublin, Ireland, June 2010.

[2] ISO/IEC DTR 8802-1, "Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Technical Reports and Guidelines – Part 1: Overview of Local Area Network Standards," 3rd ed., ISO/IEC, 1999. Available: http://www.ieee802.org/1/files/public/docs2000/J1n6125.pdf.

[3] Rockwell Automation, *Ethernet Design Considerations for Control System Networks: An Introduction*, November 2007. Available: http://samplecode.rockwellautomation.com/idc/groups/literature/documents/so/enet-so001_-en-e.pdf.

[4] G. Platt, *Process Control: A Primer for the Nonspecialist and the Newcomer*, 2nd ed., ISA, 1998.

[5] M. T. Hoske, "High Performance Industrial Networks," *Control Engineering*, July 2010, pp. 28–32.

[6] EtherCAT Technology Group, "EtherCAT – the Ethernet fieldbus." Available: http://ethercat.org/pdf/ethercat_e.pdf.

X. Biographies

David Dolezilek received his BSEE from Montana State University and is the technology director of Schweitzer Engineering Laboratories, Inc. He has experience in electric power protection, integration, automation, communication, control, SCADA, and EMS. He has authored numerous technical papers and continues to research innovative technology affecting the industry. David is a patented inventor and participates in numerous working groups and technical committees. He is a member of the IEEE, the IEEE Reliability Society, CIGRE working groups, and two International Electrotechnical Commission (IEC) technical committees tasked with global standardization and security of communications networks and systems in substations.

Francisco Chumbiauca is originally from Lima, Peru. He attended Universidad Peruana de Ciencias Aplicadas (Peruvian University of Applied Sciences) and received his bachelor's degree in Electrical Engineering in 2005. Francisco worked for Schneider Electric as an intern from 2004 to 2005. From 2006 to 2009, he worked for Rockwell Automation as a field support and training engineer for the Andean Region, traveling throughout Peru, Ecuador, Colombia, and Venezuela and assisting customers with startup, maintenance, and troubleshooting of Allen Bradley controllers and industrial networks. Francisco joined Schweitzer Engineering Laboratories, Inc. in July 2010 as an automation engineer.

Michael Rourke received his BSEE and Masters of Engineering in Electrical Engineering from the University of Idaho. He spent ten years working on research and development of control systems for steel and aluminum facilities and is a member of the Association of Iron and Steel Technology. He joined Schweitzer Engineering Laboratories, Inc. in 2000. Michael works on product development for automation and integration applications.