Journal of Circuits, Systems, and Computers
© World Scientific Publishing Company

# NOVEL DESIGN OF HETROGINIOUS AUTOMATION CONTROLLER BASED ON REAL-TIME DATA DISTRIBUTION SERVICE MIDDLEWARE TO AVOID OBSOLESCENCE CHALLENGES

SADIQ M. SAIT

*Center for Communications and Information Technology Research, and Computer Engineering Department,*
*King Fahd University of Petroleum and Minerals*
*Dhahran, 31261, Saudi Arabia*
*sadiq@kfupm.edu.sa*

Ghalib A. Al-Hashim

*Saudi Arabian Oil Company*
*Dhahran, 31311, Saudi Arabia*
*ghalib.hashim@aramco.com.sa*

Received (11 May 2015)
Revised (3 February 2016)
Accepted (14 March 2016)

Oil and gas processing facilities utilize various process automation systems with proprietary controllers. As the systems age; older technologies become obsolete resulting in frequent premature capital investments to sustain their operation.

This paper presents a new design of automation controller to provide inherent mechanisms for upgrades and/or partial replacement of any obsolete components without obligation for a complete system replacement throughout the expected life cycle of the processing facilities.

The input/output racks are physically and logically decoupled from the controller by converting them into distributed autonomous process interface systems. The proprietary input/output communication between the conventional controller CPU and the associated input/output racks is replaced with standard real-time data distribution service middleware for providing seamless cross-vendor interoperable communication between the controller and the distributed autonomous process interface systems. The objective this change is to allow flexibility of supply for all controller's subcomponents from multiple vendors to safe guard against premature automation obsolescence challenges.

Detailed performance analysis was conducted to evaluate the viability of using the standard real-time data distribution service middleware technology in the design of automation controller to replace the proprietary input/output communication. The key simulation measurements to demonstrate its performance sustainability while growing in controller's size based on the number of input/output signals, are communication latency, variation in packets delays, and communication throughput. The overall performance results confirm the viability of the new proposal as the basis for designing cost effective evergreen process automation solutions that would result in optimum total cost of ownership capital investment throughout the systems' life span. The only limiting factor is the selected network infrastructure.

*Keywords*: Automation; Controller; Real-Time; Data Distribution Service Middleware.

*List of Abbreviations:*

| | | | |
|---|---|---|---|
| **ARC** | : Automation Research Corporation | **I/O** | : Input/Output |
| **CCR** | : Centralized Control Room | **OMG** | : The Object Management Group |
| **COTS** | : Commercial Off-The-Shelf | **PC** | : Personal Computer |
| **CPU** | : Central Processing Unit | **PIB** | : Process Interface Building |
| **DCS** | : Distributed Control System | **PLC** | : Programmable Logic Controller |
| **DDS** | : Data Distribution Service | **QoS** | : Quality of Service |
| **HMI** | : Human Machine Interface | **RBD** | : Reliability Block Diagram |

## 1. Introduction

For the past forty years, the development of process automation systems including programmable logic controllers (PLC) has been evolving to raise productivity and enhance plant operation [28]. Although each system has its own unique history and characteristics, they all share a basic workload objective of acquiring process data and controlling disparate machines in the plant to work in a cohesive fashion for improved safety, higher production rates, more efficient use of materials, and better consistency of product quality. Their fundamental architecture has advanced from large centralized system with all control hardware and input/output (I/O) racks mounted in large cabinets located in the central control room (CCR) to highly distributed systems [12].

Such systems typically have limited useful lives measured by the competitive advantage they deliver and the users have always struggled with determining their expected life spans. On top of this, large oil companies are making their revenue from their production, so shutting down the plant to replace the automation system due to premature obsolescence imposes a major challenge and definitely not a preferred solution.

In 2012, Automation Research Corporation (ARC) Advisory Group performed a survey to determine the current state of the automation industry and best practices for managing the lifecycle of process automation systems from cradle to grave. There were 282 respondents from various parts of the process control industry including end users, suppliers, OEM manufacturers, and system integrators. The survey estimated the magnitude of the installed base assets of obsolete automation technology to be in the range of 65 billion US dollars. Extending the life cycle of these systems through incremental upgrades or migration paths would reduce the impact of obsolescence challenges to a limited extent. However, continuing to invest in proprietary solution is not economically attractive since similar obsolescence challenges will occur again at the end of the overall system lifecycle. Hence, this survey concluded that the best practice for safe guarding against premature automation obsolescence is to avoid proprietary solution as much as possible by capitalizing on interoperable commercial-off-the-shelf (COTS), open source, and/or multi-supplier technologies [2].

Responding to users' demand for a standard solution, automation vendors gradually started to incorporate COTS components in their automation solutions. For example, Yokogawa, Honeywell, ABB, Emerson Process Management, Rockwell Automation, Invensys, Siemens, and others have incorporated some COTS components in their latest automation solutions to ensure cost comparative advantages. However, the majority of the total automation solution including the controllers is based on proprietary hardware and software resulting in the requirement of major premature capital investments to sustain its operation throughout the life span of the controlled processing equipment. For example, the replacement of an obsolete controller would mandate the complete replacement of associated I/O racks regardless of their good condition and availability of spare parts as well as technical support. The cost of these robust and supported I/O racks is more than three times the cost of the associated obsolete controller. As a result, such a premature capital investment is very significant and increases exponentially with the number of obsolete controllers in the automation solution [7].

The objective of this paper is to present a proposal for a new vendor independent and evergreen automation controller based on multi-supplier COTS components. The main concept of this proposal is to physically and logically *decouple the controller from the I/O racks* and capitalize on emerging real-time data distribution service (DDS) middleware technology for exchanging data between them in order to realize components' interoperability.

The remainder of this paper is organized as follows. A brief discussion on the background of the problem is given in Section 2. An overview of related work is discussed in Section 3. Section 4 establishes the motivation and formulates the business case of the new proposal. The design evolution and the architecture of the proposed solution are described in Section 5. The research methodology and performance analysis is detailed in Section 6. Finally, we conclude in Section 7.

## 2. Background

In today's competitive production environment, there is a very high demand on process industries to economically mass produce many customized products rather than a single product. This challenging requirement mandates the utilization of latest agile and flexible PLC technologies to increase productivity, reliability, and quality while minimizing cost.

A PLC consists of components illustrated in Figure 1: central processing unit (CPU), memory, input modules, output modules, and power supply.

Four basic steps, in the operation of all PLCs, are executed continually in a repeating loop as shown in Figure 2; input scan, program scan, output scan, and housekeeping. During the input scan, the PLC input modules detect the state of all process input measurement devices. During the program scan, the PLC CPU executes the user created program logic based on the input data. During the output scan, the PLC output modules energize or de-energize output devices based on the result of the executed program logic. During the housekeeping step, the PLC CPU performs internal diagnostics and communications with programming terminals and/or HMI consoles. The minimum PLC scan time resolution is 100ms. This scan time includes 20ms dedicated for scanning the input modules, 50ms dedicated for solving the control logic, 15ms dedicated for updating the output modules, and 15ms dedicated for diagnostics and housekeeping [26].

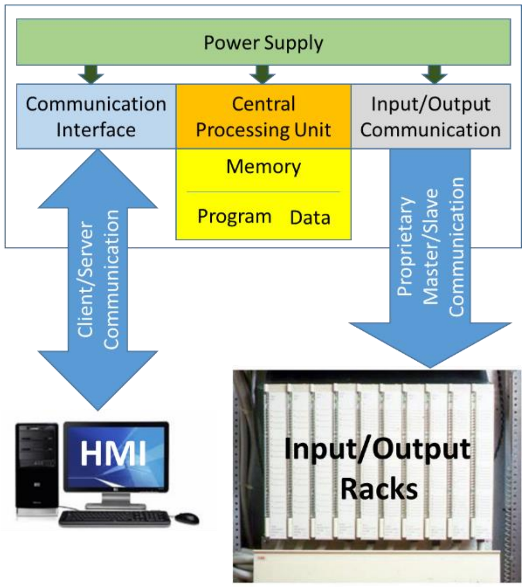


Figure 1 - PLC Hardware Block Diagram

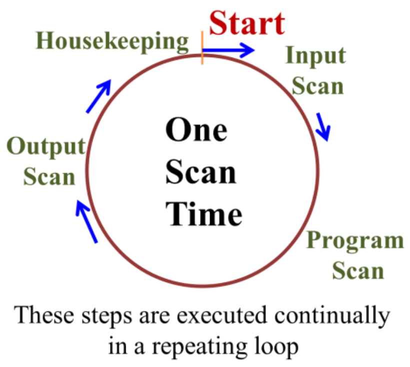


Figure 2 - PLC Operation

Figure 3 shows a typical layout of PLC-based system architectures. Included in this figure is an illustration of hardwired connections from the field instruments to the operation human machine interface (HMI) consoles located at the CCR spanning through field junction boxes, marshaling cabinets, I/O systems cabinets, controller system cabinets, process interface building (PIB), and CCR rack room.

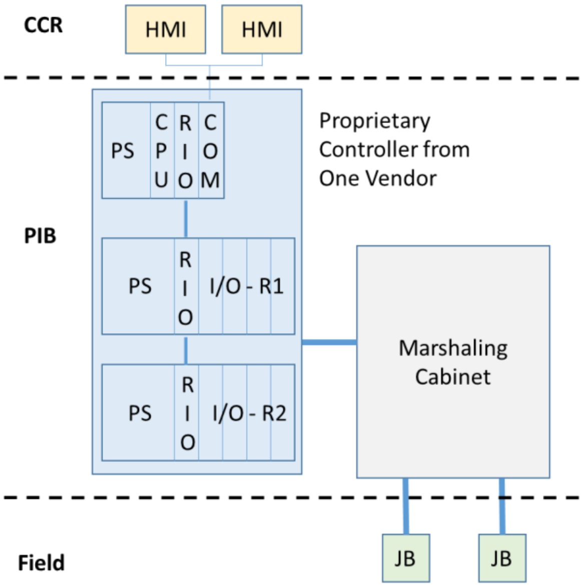


Figure 3 - Typical PLC-Based System Architecture

The junction boxes are enclosures used for cables interconnection between field devices and marshaling cabinets in PIBs. The function of the marshaling cabinet is to interface the incoming multicore cables with the I/O module system cables and to perform the cross wiring function. Cross wiring is always necessary due to three reasons; (1) the requirement for routing the input and output signals to the designated PLC, (2) the mix of input and output field signals within the same incoming multicore cables and the requirement to split them into consecutive and dedicated terminals for the associated I/O modules terminals, and (3) the number of incoming field signals within multicore cables is often different from the number of channels per I/O module. The purpose of the system cabinet is to provide terminals to interface with the marshaling cabinets and to house the PLC power supply, I/O racks, controller CPU, communication modules, engineering work station, and the auxiliary power supply for powering the field instruments.

The PIB is an explosive proof building used to house and protect the system and marshaling cabinets from deteriorating effects of the weather. It is unmanned and environmentally controlled building suitable to house delicate active electronics. Its location is carefully selected to withstand any unexpected field process explosion and is as close as possible to large number of junction boxes in order to minimize cabling.

The CCR is a room serving as a central space where a large physical facility or physically dispersed service can be monitored and controlled. The CCR for vital facilities are typically tightly secured and manned continuously to ensure unceasing vigilance. It has two major sections: rack room and control room. The function of the rack room is identical to the function of the PIB. The control room includes HMI, engineering, and maintenance consoles, as well as an auxiliary control panel for hardwired pull/push buttons for emergency and plant shutdown [10].

Oil and gas processing facilities are highly susceptible to intrusion and cyber-attacks. The PLC design includes four compulsory security protection mechanisms to ensure enough protection against potential malicious attacks: (1) communication between the controller and associated I/O racks are physically isolated from any other communication networks, (2) PLC equipment is located in a physically secured PIB with key lock, (3) PLC equipment is physically enclosed in secured system cabinets in the PIB with key locks, and (4) PLC CPU module includes a physical key switch with three positions as shown in Figure 4 (Memory Protect ON, Data Change, and Memory protect OFF).

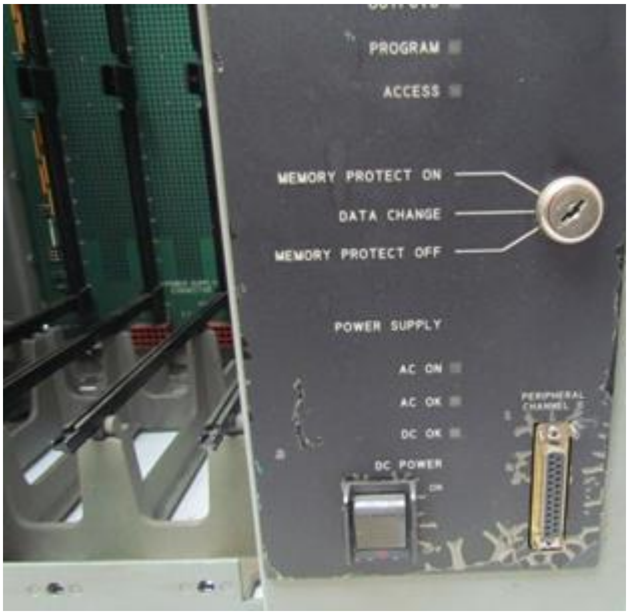


Figure 4 - Typical PLC CPU Security Protection Key Switch

## 3. Related Work

The lifecycle management of process automation systems has improved significantly utilizing distributed architectures; however, it is still an ever-increasing issue for sustaining the automation capital investment of hydrocarbon processors and other process manufacturers [20]. Recently, through vendors' user steering committees, major automation consumers started to leverage their voting privileges to influence vendors' research and development selection process to incorporate COTS components in the automation solution everyplace feasible. The following section describes latest systems development for major automation vendors.

## 3.1. *Utilization of COTS Components in Automation Systems*

Yokogawa has adopted state-of-the-art technologies to develop the latest distributed control system (DCS) system called CENTUM VP offering customers the convenience of using COTS hardware for the HMI workstations based on personal computers (PC) and Microsoft Windows operating system [17]. Honeywell, on the other hand, has patented distributed system architecture technology offering users with seamless operation data integration of multiple Honeywell Experion PKS distributed control systems utilizing virtual fault tolerant communication networks based on COTS Ethernet components [3]. ABB's latest DCS solution has evolved using "Aspect Object" technology and client-server architecture to streamline controller communications and leverage visualization technology at the HMI layer based on COTS server and workstation computer hardware [1]. Emerson Process Management has evolved the latest DCS system capitalizing on COTS technologies at the HMI layer and associated communication networks, and then added application functionality to allow the equipment to function like other parts of the automation system including plug-and-play capability, full lifecycle support without upgrades, and built in security [4]. Rockwell Automation is applying control and visualization solutions and COTS technologies to build multifaceted systems that ensure effective interaction of all processes, without degrading mission critical functions [22]. Foxboro Evo DCS system from Invensys has evolved to power the enterprise control system. It features a component object-based platform with common industrial service-oriented architecture. So users get the opportunity to expand from process control to enterprise control with a single system based on COTS technologies [11].

The use of COTS technologies was driven by DCS users and was then considered to be the best solution that provides tremendous increase in functionality and cost advantages to manufacturers of today's automation systems. However, the programmable controllers are still proprietary. A primary enabler for designing programmable controllers based on COTS components is the emerging real-time reliable and fault tolerant data-centric middleware technology. This technology, detailed in the next section, provides seamless cross-vendor interoperability.

## 3.2. *Real-Time DDS Data-Centric Middleware*

Middleware is a collection of technologies and services to enable the integration of subsystems and applications across an overall system. Several standardization efforts in many aspects of middleware technologies resulted in different classifications of the middleware solutions. A broad approach middleware classification based on the types of heterogeneity including platform, programming language and communication connectivity is described in [23]. Another paper classified the middleware based on its capabilities in meeting non-functional requirements including capabilities to provide communication among heterogeneous components, to extend its functionalities using open interface, to sustain system performance with higher loads in future, to recover from hardware or software failures, to provide security policies and mechanisms, to guarantee quality of service (QoS) for real-time applications, and to cope with changes in the applications and/or users requirements [5].

Many architectural models used in the development of middleware systems are found in literature. Most of the architectures have evolved from point-to-point, client-server, and publish-subscribe models.

Point-to-point is the simplest tightly coupled form of communication. TCP is a point-to-point network protocol designed in the 1970s. While it provides reliable, high bandwidth communication, TCP is cumbersome for systems with many communicating nodes [8].

To address the scalability issues of the Point-to-Point model, developers turned to the client-server model for centralized information and distributed applications, and many other paradigms are built upon it. However, if information is being generated at multiple nodes, client-server model is inefficient and precludes deterministic communications, since the client does not know when new information is available [27].

A solution to the above limitation on client-server models for real-time systems where information is being generated at multiple nodes is to adopt publish-subscribe communication model. In this model, computer applications subscribe to data they need and publish data they want to share. Messages pass directly between the publisher and the subscribers, rather than moving into and out of a centralized server. Most time-sensitive information intended to reach many people is sent by publish-subscribe systems. Examples of publish-subscribe systems in everyday life include television, magazines, and newspapers. This direct and simultaneous communication among a variety of nodes makes publish-subscribe network architecture the best choice for systems with complex time-critical data flows, even in the presence of unreliable delivery mechanisms [18].

### 3.2.1. *Latest Development in Publish-Subscribe Middleware*

One of the most important efforts to standardize publish-subscribe middleware is the development of DDS specification by Object Management Group, Inc. (OMG). Data-centric publish-subscribe standard is the portion of the OMG DDS specification that addresses the specific needs of real-time data-critical applications and describes the fundamental concept supported by the design of the application programming interface. It focuses on the distribution of data between communicating applications, and provides several mechanisms that allow application developers to control how communication works and how the middleware handles resource limitations and error conditions. The communication is based on passing data of known types in named streams from publishers to subscribers. In contrast, in object-centric communications the fundamental concept is the interface between the applications. [14,15,25].

An object-centric system consists of interface servers and interface clients, and communications are based on clients invoking methods on named interfaces that are serviced by the corresponding server. Data-centric and object-centric communications are complementary paradigms in a distributed system. Applications may require both. However, real-time communication often fits a data-centric model more naturally. For example, real-time automation control systems often requires specific features including efficiency, determinism, flexibility delivery bandwidth, and fault-tolerant operation [13,19].

***Efficiency:*** Real-time systems require efficient data collection and delivery. Only minimal delays should be introduced into the critical data-transfer path. Publish-subscribe model is more efficient than client-server model in both latency and bandwidth for periodic data exchange. Publish-subscribe architecture greatly reduces the overhead required to send data over the network compared to client-server architecture. Occasional subscription requests, at low bandwidth, replace numerous high-bandwidth client requests. Latency is also reduced, since the outgoing request message time is eliminated. As soon as a new publication data sample becomes available, it is sent to the corresponding subscriptions [6].

***Determinism:*** Real-time automation applications also care about the determinism of delivering periodic data as well as the latency of delivering event data. Once buffers are introduced into a data stream to support reliable connections, new data may be held undelivered for an unpredictable amount of time while waiting for confirmation that old data was received. Since publish-subscribe does not inherently require reliable connections, implementations can provide configurable trade-offs between the deterministic delivery of new data and the reliable delivery of all data [9].

***Flexibility Delivery Bandwidth:*** Typical real-time control systems include both real-time and non-real-time nodes. The bandwidth requirements for these nodes are different. For example, an application may be sending data samples faster than a non-real-time application is capable of handling. However, a real-time application may want the same data as fast as it is produced. Data-centric publish-subscribe allows subscribers to the same data to set individual limits on how fast data should be delivered to each subscriber. This is similar to how some people get a newspaper every day while others can subscribe to only the Friday paper [16].

***Fault-Tolerant Operation:*** Real-time automation applications are required to run in the presence of component failures. Often, those systems are safety critical, or carry financial penalties for loss of service. The applications running in those systems are usually designed to be fault-tolerant using redundant hardware and software. Backup applications are often “hot” and interconnected to primary systems so that they can take over as soon as a failure is detected. Publish-subscribe model is capable of supporting many-to-many connectivity with redundant publishers and subscribers. This feature is ideal for constructing fault-tolerant or high availability applications with redundant nodes and robust fault detection and handling services [21].

***Real-Time DDS Interoperability:*** With the increasing adoption of DDS in large distributed systems, it was desirable to define a standard *wire protocol* that allows DDS implementations from multiple vendors to interoperate. Hence, OMG developed the real-time DDS interoperability wire protocol specification to ensure that information published on a topic using one vendor's DDS implementation is consumable by one or more subscribers using different vendor's DDS implementations. The DDS wire protocol is capable of taking advantage of the quality of service settings configurable by DDS to optimize its use of the underlying transport capabilities. In particular, it is capable of exploiting the multicast, best-effort, and connectionless nature of many of the DDS quality of service settings [24].

## 4. Motivation

The conventional controllers are based on a monolithic architecture in which functionally distinguishable aspects such as the I/O racks, the main control module, and the control application, are not architecturally separate standard components but are all proprietary and interwoven. This is similar to the mainframe computer architecture of the past. This architecture does not allow for changing the design of certain aspects of the controller easily without having to overhaul the entire control application or to buy another controller altogether. At the same time, there are no compatible alternatives from other vendors to replace obsolete proprietary components. For example, this happens when the controller becomes obsolete while the associated I/O racks are still current and can be supported for the next 20 years. A premature capital intensive investment, about 75% of the total cost of ownership, is required for the replacement of the I/O racks in order to replace the obsolete controller. Hence, retaining the current I/O racks, when replacing the associated obsolete controller due to either the shortage of third party critical and proprietary subcomponents or the adoption of a new marketing strategy by the vendor to surpass its competitors and/or to increase its market share, is economically very attractive. However, the user will not be able to capture this opportunity due to the monolithic proprietary controller architecture where a complete replacement becomes compulsory.

To address the escalating capital investment challenges facing the oil and gas industries, and others, in relation to process automation obsolescence and life cycle management, two features must be accomplished; moving away from the monolithic controller architecture and utilizing multi-supplier COTS components.

## 5. Proposed Solution

In this section, we discuss the evolution of the proposed changes in design of the conventional automation controller.

A modular system can be characterized by functional partitioning into discrete scalable and reusable modules, rigorous use of well-defined modular interfaces, and making use of industry standards for interfaces. Besides reduction in cost due to lesser customization, and less learning time, and flexibility in design, modularity offers other benefits such as exclusion of obsolete modules and augmentation by merely plugging in different current modules. In the computer hardware industry, this idea allowed building computers with easily replaceable parts that use standardized interfaces and allowed upgrading or replacing obsolete aspects of the computer easily without having to buy another computer altogether.

Similarly, we propose to solve the current monolithic controllers' architecture problem by physically and logically decoupling the I/O racks from the main control module and converting them into distributed autonomous process interface systems. For this concept to work properly, real-time reliable and fault tolerant publish-subscribe data-centric middleware is required for providing seamless cross-vendor interoperable communication between the controller and the distributed autonomous process interface systems as shown in Figure 5.

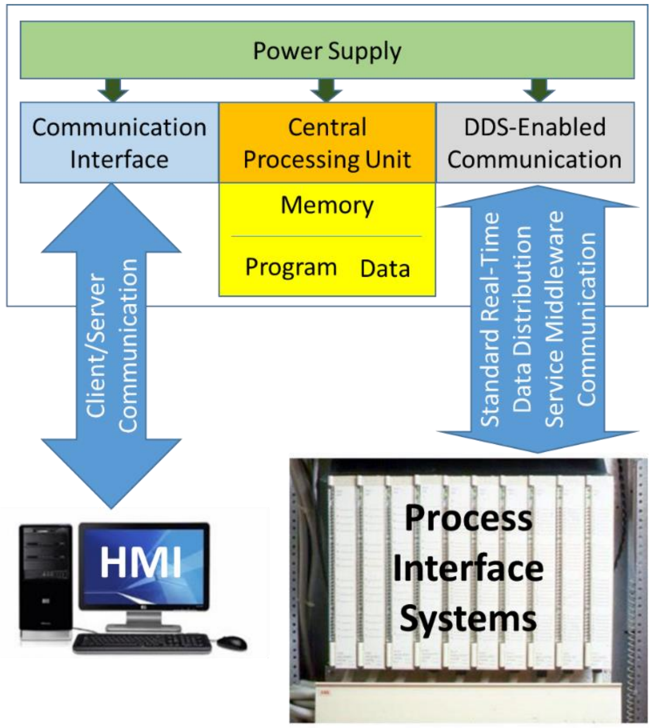

Power Supply
Communication Interface
Central Processing Unit
DDS-Enabled Communication
Memory
Program
Data
Client/Server Communication
Standard Real-Time Data Distribution Service Middleware Communication
HMI
Process Interface Systems

Figure 5 - Heterogeneous Controller Hardware Block Diagram

The OMG DDS is the first open international middleware standards suitable for addressing publish-subscribe communications for real-time data-critical applications and embedded systems. The above modularity concept results in the development of an evergreen automation solution based on a modular architecture and vendor independent components that can last for the expected life span of the controlled processing facilities. The life cycle of this automation solution can be managed and sustained using a replacement on failure strategy based on multi-supplier COTS components without the need to buy another automation controller altogether.

The proposed heterogeneous automation controller includes six compulsory security protection mechanisms: (1) Communication between the controller and associated process interface systems are physically isolated from any other communication networks, (2) Controller equipment is located in a physically secured process interface building with key lock, (3) Controller equipment is physically enclosed in secured system cabinets in the secured process interface building with key locks, (4) Controller CPU module has a physical key switch with three positions as (Memory Protect ON, Data Change, and Memory Protect Off), (5) Each process interface system has a physical key switch with three positions as (Memory Protect ON, Data Change, and Memory Protect Off), (6) Communication between the controller and associated process interface systems are secured based on an exclusive publish/subscribe relationship for each data point.

## 5.1. *Comparison between the Conventional and New Control Systems*

The following is a detailed example to illustrate the difference between the conventional and the proposed heterogeneous controller architectures.

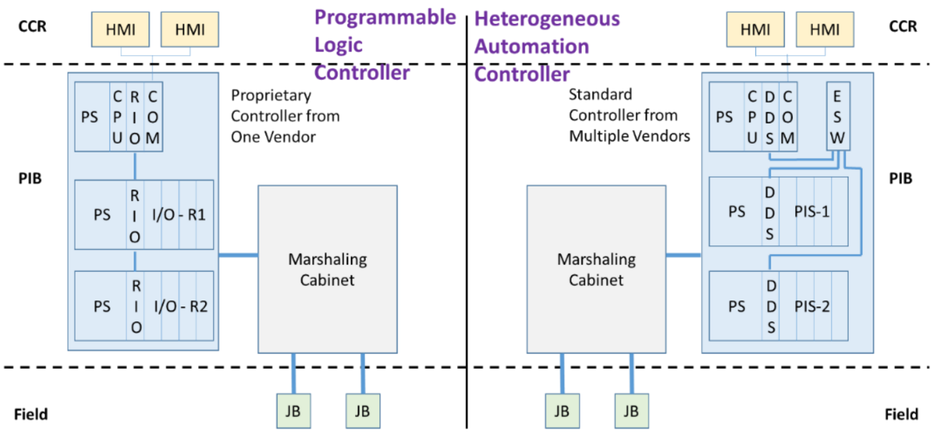


Figure 6 - An Example of One Controller with Two Input/Output Racks

The Left side of Figure 6 shows a typical conventional PLC system architecture based on a proprietary or Ethernet-based local area control network, located in the CCR to connect two HMI consoles and one controller through one control segment extended to the PIB, located close to the processing facilities. The proprietary controller is connected to its associated I/O racks using a proprietary daisy-chained remote I/O communication links.

The equivalent controller architecture based on the proposed concept is shown in the right side of Figure 6. The new controller architecture consists of three main components empowered by real-time reliable and fault- tolerant DDS middleware; controller, standard Ethernet communication equipment, and distributed autonomous process interface I/O systems. The process interface systems are autonomous because they are self-contained and could run without the need for a controller specifically for process monitoring.

The difference between the Master/Slave communication and the DDS Publish/Subscribe communication is very significant. For the Master/Slave communication, the Master is the Controller and the Slaves are the I/O Racks. Every 80ms during the input scan, the controller pulls the latest input status from the I/O racks sequentially. Every 85ms during the output scan, the controller pushes the final output states to the I/O racks sequentially. For the DDS Publish/Subscribe communication, the controller as well as the process input systems are all Masters. The Master publishes the data when there is any changes in the states in real-time and the DDS Middleware immediately distributes the changes in data to the subscribers. As a result, the controller will have the latest up to date input data from the controller DDS communication module during the beginning of every operation scan. Also, the controller will update the controller DDS communication module with any changes in the output states immediately after the completion of the program scan. After that, the controller DDS communication module publishes all changes to the subscribers through the DDS middleware. Hence, there are two operation cycles running in parallel as shown in Figure 7, one for the controller and another one for each process interface system. Therefore, the typical 100ms controller operation scan is divided into 1ms for updating the input image table, 83ms dedicated for solving the control logic, 1ms for updating the output image table, and 15ms dedicated for the diagnostics and housekeeping.

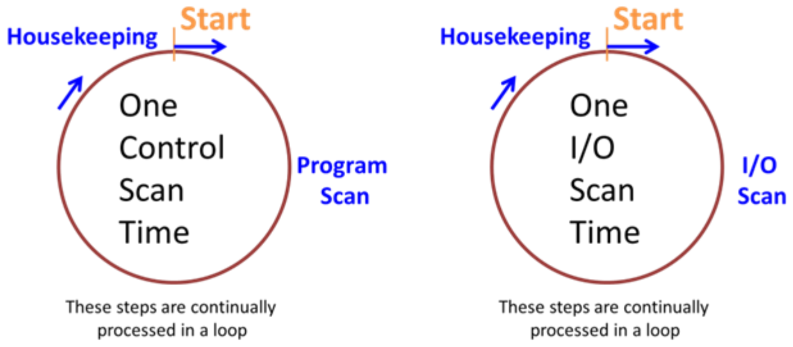

Housekeeping
Start
One Control Scan Time
Program Scan
These steps are continually processed in a loop
Housekeeping
Start
One I/O Scan Time
I/O Scan
These steps are continually processed in a loop

Figure 7 - Heterogeneous Automation Controller Operation

As a result, the processing time for the logic is increased to provide enough processing time for 66% additional control logic for the application without affecting the overall 100ms scan resolution. If the program size is kept the same, the diagnostic and housekeeping resolution for the controller is reduced from 85ms to 51ms, equivalent to 40% reduction in waiting time for executing the diagnostics script. Also, the diagnostics for the process interface systems are done in parallel and identified in real-time similar to the process data within 1ms. Therefore, the improvement in diagnosing the I/O racks is reduced from 85ms to 1ms, equivalent to 98% reduction in waiting time for executing the diagnostics script. The overall conclusion that the proposed controller is far faster in identifying faults within the system than the current PLC.

The PLC includes a simplex proprietary master/slave communication protocol for communicating with the daisy-chained I/O racks. This proprietary protocol does not support redundancy. Therefore, any faults in the RIO communication hardware will jeopardize the whole controller. However, the communication among the heterogeneous automation controller and associated process interface systems is a standard Ethernet based DDS communication infrastructure that can be either simplex, parallel, or fault-tolerant communication as shown in Figure 8. The simplex architecture can be single Ethernet Switch or a multi-drop Ethernet Bus which is equivalent to the proprietary daisy-chained communication network structure within a PLC. However, the parallel structure is a single-fault tolerant architecture with two disjoint paths from the controller to any of the process interface systems.

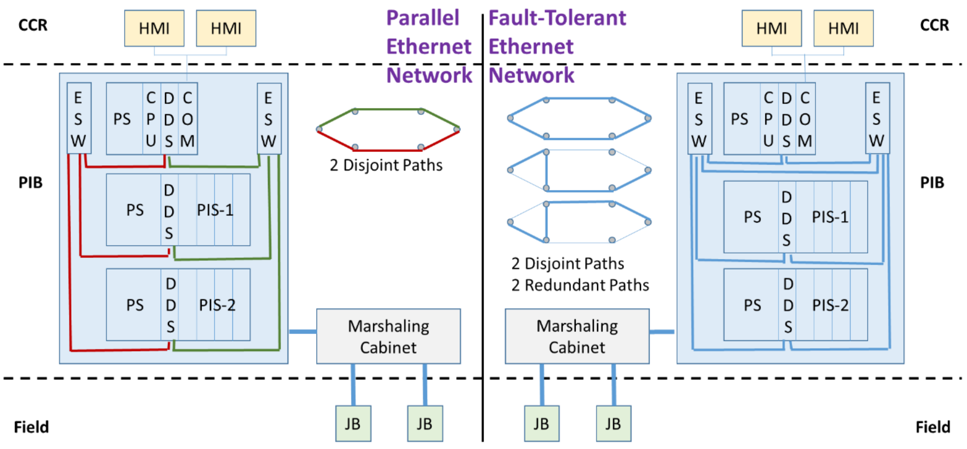


Figure 8 - Parallel and Fault-Tolerant DDS-Enabled Ethernet Networks

The communication among the controller and the associated process interface systems is either using the green leg including the Ethernet Switch and associated link segments or the red leg tolerating any single communication failures. One leg is considered the primary communication network and the other leg is the backup communication network. The fault-tolerant structure is a multi-fault tolerant architecture with two disjoint paths as well as two redundant paths from the controller to any of the associated process interface systems as shown in Figure 8. The fault-tolerant communication network allows the continuous operation of the Ethernet Switch while there is a failure in any associated link segments.

## 5.2. *Network Reliability Analysis*

The terminal reliability of communication network is the probability of having at least one available path between any sources to any destinations. The terminal reliability is calculated using the combinatorial series and parallel models assuming that the failure probabilities of different components of the system are independent. For the parallel Ethernet network, it is assumed that no communication packets can be routed through a faulty Ethernet switch, and all links connected to a faulty switch are useless. This will result in disjoint communication paths from a source to a destination. For the fault-tolerant Ethernet network, it is assumed that no communication frames can be routed through a faulty Ethernet link, and all other communication links associated with the same Ethernet switch are useful. Under the above conditions, the following formulas describing the terminal reliability of the systems are developed. Sometimes a “success” diagram is used to describe the operational models of a system network from a source to a destination.

Figure 9(a) shows the success diagram for the fault-tolerant Ethernet I/O network. If the success diagram becomes too complex to evaluate exactly, upper-limit approximation on the network terminal reliability can be used. An upper bound on terminal reliability is $R \leq (1 - \prod_{i=1}^{i=RP}(1 - R_{path-i}))$ where RP is the number of redundant paths available from a source to a destination and $R_{path-i}$ is the serial reliability of path-i. The upper bound on terminal reliability calculated as if all paths were in parallel. This calculation is an upper bound because the paths are not independent. That is the failure of a single networking element affects more than one path. Therefore, this approximation gets closer to the actual terminal reliability when terminal reliability of a path is small. Placing the paths in parallel yields a reliability block diagram (RBD).

Figure 9(b) shows the RBD for the success diagram of the fault-tolerant Ethernet I/O network shown in Figure 9(a). Using the combinatorial series and parallel models, the upper bound on terminal reliability of the fault-tolerant Ethernet I/O network is calculated as follows $R \leq (1 - \prod_{i=1}^{i=4}(1 - R_{path-i}))$ where $R_{path-1,path-4} = R_{link}^{4}R_{Switch}^{3}$ and $R_{path-2,path-3} = R_{link}^{5}R_{Switch}^{4}$ assuming that the reliability of all communication links are equal and represented by $R_{link}$ and the reliability of all Ethernet switches are equal and represented by $R_{Switch}$. Therefore, the upper bound on terminal reliability of the fault-tolerant Ethernet I/O network = $R_{level-1} \leq 1 - (1 - R_{link}^4 R_{Switch}^3)^2(1 - R_{link}^5 R_{Switch}^4)^2$.

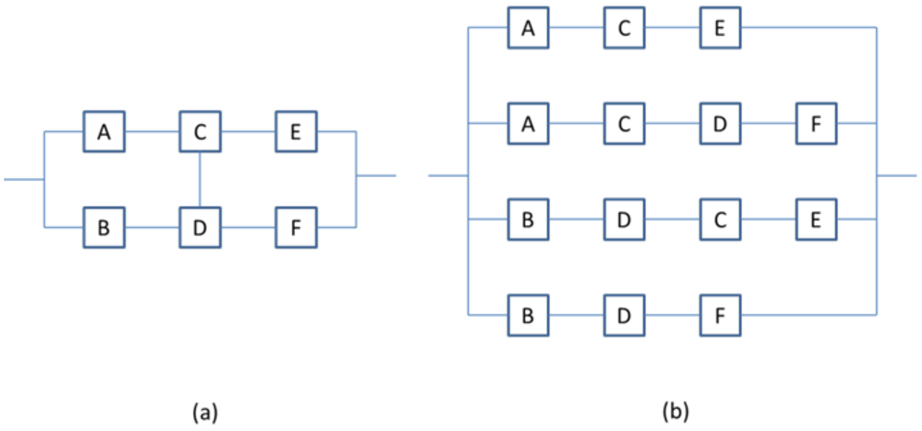


Figure 9 - (a) Success Diagram, (b) Reliability Block Diagram

Figure 10 shows the communication network terminal reliability comparison between the PLC and the heterogeneous automation controller with parallel and fault-tolerant networks.

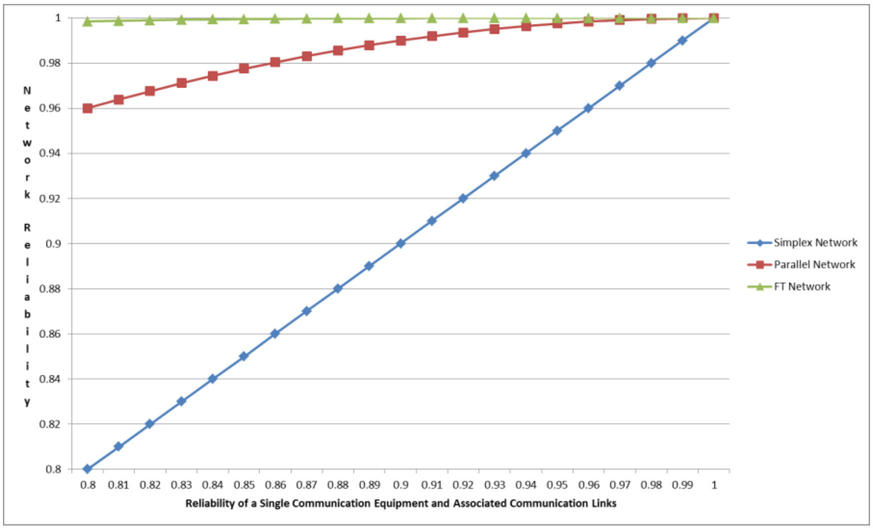


Figure 10 - Network Terminal Reliability of Parallel and Fault-Tolerant Networks

### 5.3. *Life Cycle Cost Analysis*

The life cycle cost of a control system includes the initial capital cost and the annual capital cost to sustain the system operation throughout the life span of the processing facility. The initial capital cost includes the design phase, detailed engineering phase, implementation phase, installation phase, and commissioning and startup phase. The annual capital cost includes the cost for spare parts, technical support, training, and the implementation of any required hardware and/or software revisions.

The following are the economic model assumptions:

- The life span of the processing facility is 45 years.
- The life span of the proprietary PLC control system is 15 years.
- For the proprietary PLC solution, a total control systems replacement is required every 15 years. Therefore, 2 complete control systems replacements are required within the life span of the processing facility, during the 15th year and during the 30th year.
- The cost escalation factor is 1% every year.
- The total initial capital cost of a control system with 25,000 input/output signals is 25 million US dollars, average of US$ 1000 per I/O signal.
- The annual cost of the contract to manage and maintain the control system for sustaining the processing operation is 0.5% of the initial cost of the control systems, subject to the annual escalation cost factor. Therefore, the contract cost for the first year is US$ 125,000 and for the second year is US$ 126,250 due to the incremental cost escalation.

The total cost of the control systems over the life span of the processing facility based on the proprietary PLC is US$ 94,159,587 and based on the proposed heterogeneous automation controller is US$ 32,060,134. Using the new standard solution to avoid obsolescence challenges results in an approximately 66% cost saving throughout the life span of the processing facility.

## 6. Performance Analysis

The proposed automation controller has been evaluated empirically using software based simulation model to demonstrate its performance sustainability while growing in size based on the number of I/O signals.

## 6.1. *Performance Test Setup*

The model set up, shown in Figure 11, includes one 2.1GHz Lenovo i7-4600U Thinkpad, three 2GHz Lenovo i7-3667 Thinkpads, and one 16-port 10/100 Mbps fast Ethernet switch. Real-time Connext DDS professional middleware version 5.1.0.14-i86Win32VS2013 from Real-Time Innovations, Inc. is installed in all Lenovo Thinkpad laptops. The four laptops are connected to one 16-port 10/100 Mbps fast Ethernet switch.

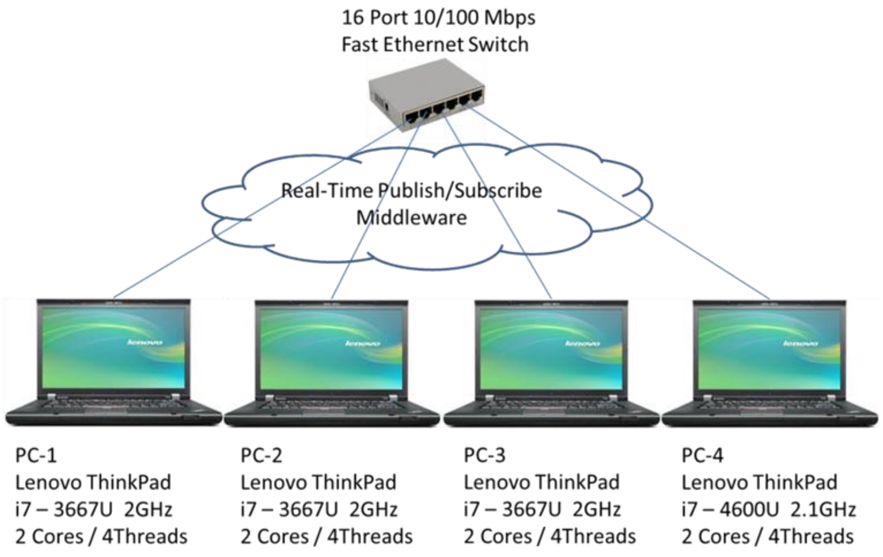


Figure 11 – COTS-Based Automation Controller

## 6.2. *DDS Quality of Service Polices for Automation Control*

DDS QoS policies for real-time systems can be used to control and optimize network as well as computing resource to ensure that the right information is delivered to the right subscriber at the right time. The default values are used with the following exceptions to meet the requirement of the automation controller design.

***Reliability:*** The reliability QoS policy indicates the level of guarantee offered by the DDS in delivering data to subscribers. Possible variants are *reliable* and *best effort*. With the selection of *reliable* parameter in steady-state, the middleware guarantees that all samples in the publisher history will eventually be delivered to all the subscribers. However, the *best effort* parameter indicates that it is acceptable to not retry propagation of any samples. The *reliable* option is selected in this experiment.

***Durability:*** The durability QoS policy controls the data availability with respect to late joining publishers and subscribers; specifically the DDS provides the following variants: *volatile, transient local, transient,* and *persistent*. With *volatile* option, there is no need to keep data instances for late joining subscriber. With *transient local* option, the data instance availability for late joining subscriber is tied to the publisher availability. With *transient* option, the data instance availability outlives the publisher. With the *persistent* option, the data instance availability outlives the system restarts. The durability service QoS policy is used to configure the history QoS policy and the resource limits QoS policy used by the fictitious subscriber and publisher used by the *persistence service*, responsible for implementing the durability QoS policy options of *transient* and *persistence*. The *persistent* option is selected in this experiment.

***History:*** The history QoS policy controls whether the DDS should deliver only the most recent value, attempt to deliver all intermediate values, or do something in between. The policy can be configured to provide the following semantics for how many data samples it should keep: *keep last* and *keep all*. With *keep last* option, the DDS will only attempt to keep the most recent depth samples of each instance of data identified by its key. However, with the *keep all* option, the DDS will attempt to keep all the samples of each instance of data identified by its key. The *keep all* option is selected in this experiment.

***Ownership:*** The ownership QoS policy specifies whether it is allowed for multiple publishers to write the same instance of the data and if so, how these modifications should be arbitrated. Possible options are: *shared* and *exclusive*. With *shared* option, multiple publishers are allowed to update the same instance and all the updates are made available to the subscriber. However, the *exclusive* option indicates that each instance can only be owned by one publisher, but the owner of an instance can change dynamically due to liveliness changes and the selection of the owner is controlled by setting of the ownership strength QoS policy. The ownership strength QoS policy specifies the value of the strength used to arbitrate among publishers that attempt to modify the same data instance. The policy applies only if the ownership QoS policy is set to *exclusive*. The *exclusive* option is selected in this experiment.

## 6.3. *Performance Test Criteria*

The focus of this empirical test is to validate the viability of using real-time DDS middleware to exchange required interaction traffic between the controllers and the autonomous process interface systems for safe and reliable operation of the processing facilities. The measuring performance criteria are the average latency, transmission delay variation, and throughput. The communication test between a publisher and a subscriber is as follows. The I/O system is the publishing side and the controller is the subscribing side. The publishing side writes data, a total of 30 million data samples, to the middleware as fast as it can. Every time, after writing 1000 data samples to the middleware, it sends a special sample requesting an echo from the subscribing side. On one hand, the publishing application publishes throughput data and at the same time it also subscribes to the latency echoes. On the other hand, the subscribing applications subscribe to the throughput data, in which the echo requests are embedded; they also publish the latency echoes. The publisher uses the request for an echo exchange to measure the round-trip latency. The time stamp is logged by the publisher from the start of sending the data sample request until it receives the echo of the data sample back from the subscriber. The communication latency between a publisher and a subscriber is one half of the round-trip latency. The average communication latency between a publisher and a subscriber is the average of the 30 thousand times of latency measurement during one test. The reason for measuring the round-trip latency rather than one-way latency is to overcome the challenge of ensuring accurate clock time synchronization between the publisher and the subscriber. Each test scenario is repeated eight times with different data packet size of 100, 200 400, 800, 1600, 3200, 6400 and 12800 bytes. The change in data size represents the change in the number of I/O signals. The subscriber measures the throughput by counting the number of received data packets per second and the throughput rate of Megabits per second. Figure 11 depicts the complete automation controller architecture utilized in the performance testing. All four machines are configured with RTI real-time Connext DDS middleware. The normal minimum PLC scan time resolution is 100ms and a total of 35ms is dedicated for I/O communication services. Therefore, the average communication latency between the controller and the I/O system through the real-time publish/subscribe DDS middleware shall be within 35ms. The baseline performance test is to measure the latency and throughput of one controller and one I/O system within each laptop.

### 6.3.1. *Communication Latency Analysis*

The measured average latency for the three identical laptops as well as the fourth laptop is shown in Figure 12. The performance result of the average communication latency between the controller and the I/O system in all laptops is within 1ms, very well below the required scan time resolution while varying the controller size from 100 bytes equivalent to a PLC with 400 digital I/O and 50 analog I/O, to a controller size of 12,800 bytes equivalent to a PLC with 80,000 digital I/O and 2,800 analog I/O. The data shows that communication latency remains consistently low as message size increases. This is an excellent result showing that the real-time publish/subscribe DDS middleware was able to cope with the huge increase in data loading without any significant impact on the controller performance.

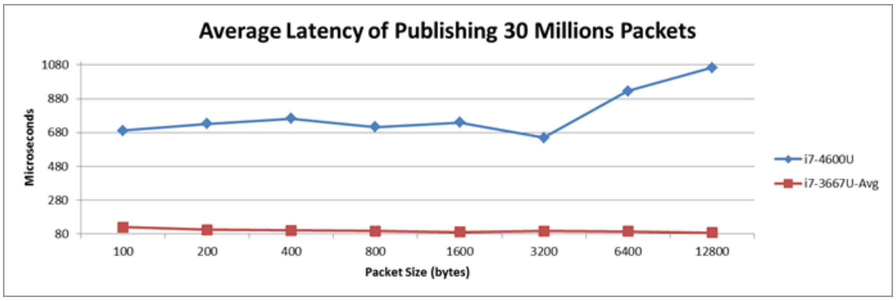


Figure 12 - Average Latency of Controller and I/O system within each PC

### 6.3.2. *Communication Latency with Jitter Analysis*

Figure 13 shows the latency with jitter analysis for the first laptop. Jitter is the variation in latency as measured in the variability over time of the packet latency across the communication medium. With constant latency, there is no variation or jitter. A system is more deterministic if it exhibits low jitter. The blue series show the minimum measured latency and green series show the 99th percentile latency. Latency at 99th percentile means that only 1% of the data samples exhibited latency larger than this value. Even at large packet sizes, the variation between the minimum and 99% latency remains consistently low. This shows that the real-time publish/subscribe DDS middleware between the controller and the I/O system exhibits very low jitter and very high determinism, making it suitable for real-time and mission-critical applications.

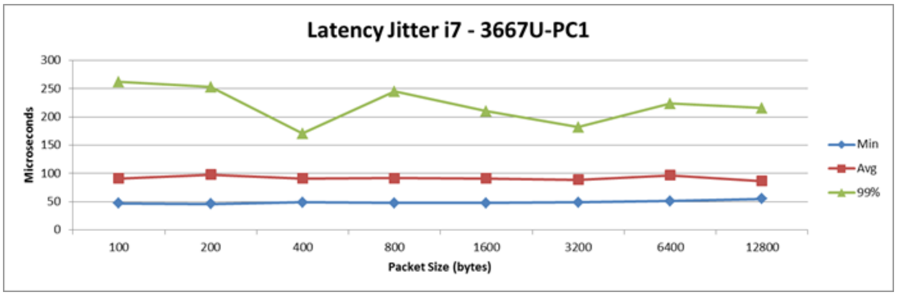


Figure 13 – Average Latency of 1 Controller and 1 I/O system within PC1

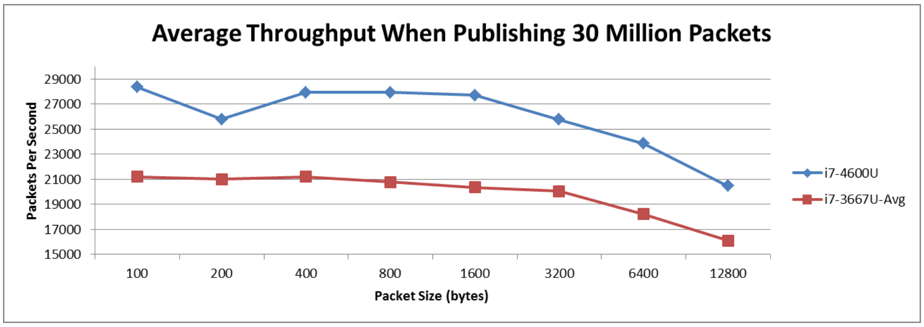
### 6.3.3. *Communication Throughput Analysis*

For the throughput analysis, the publisher sends data to one subscriber application. The performance test goes through the following phases:

1. The publisher signals the subscriber application that it will commence, and then starts its own clock. The duration of the test is based on the number of data samples to be written to the middleware; in this case it is 30 million packets.
2. The subscriber starts measuring the number of data samples received.
3. After the desired duration is over, the publisher signals the subscriber that the experiment is over. The subscriber will then divide the number of samples received by the elapsed time to report the throughput observed at the receiver.

Maximum throughput is achieved when the publisher sends as fast as the subscriber can handle messages without dropping a packet. That is, the maximum throughput is obtained somewhere between the publisher sending too slowly, not maximizing the available pipe, and the publisher swamping the subscriber, overflowing the pipe. For this reason, the test makes the publisher try a range of sending rates. For the absolute maximum throughput to be observed, the optimal sending rate must be in the range. The measured average throughput bandwidth between one controller and one I/O system for the three identical laptops as well as the fourth laptop measured in packets per second and Megabits per second is shown in Figure 14.

The graph shows sustainable publish/subscribe throughput bandwidth between one controller and one I/O system within each laptop in terms of packets per second and Megabits per second. Obviously, the slight decrease in the throughput in terms of number of packets per second is due to the increase in transmission time of each packet. However, the throughput bandwidth in terms of Megabits per second increases significantly with the increase in the size of the packet. This indicates that the real-time DDS middleware was able to cope with the huge increase in data loading and fully utilized available data bus communication bandwidth between the controller and the I/O system. In other words, it does not impose any inherent limit on the aggregate data messaging capacity, making it suitable for scalable automation platforms.



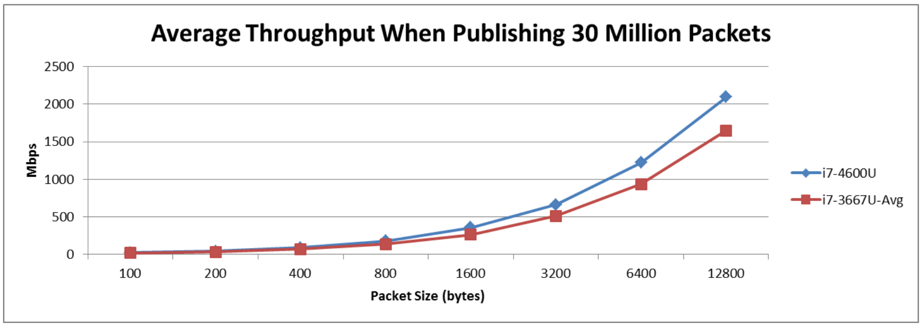


Figure 14 – Average Throughput of Controller and I/O system within each Laptop

### 6.3.4. *Impact Analysis of using Ethernet Switch between the Controller and I/O System Machines*

The set up for the next performance test configuration is to host the controller application in one laptop and to host the I/O system in another identical laptop. The communication between the controller and the I/O system is implemented through a 16-port 10/100 Mbps

3Com fast Ethernet switch using real-time publish/subscribe DDS middleware. The measured average latency and throughput in terms of packets per second and Megabits per second for the two identical laptops are shown in Figure 15.

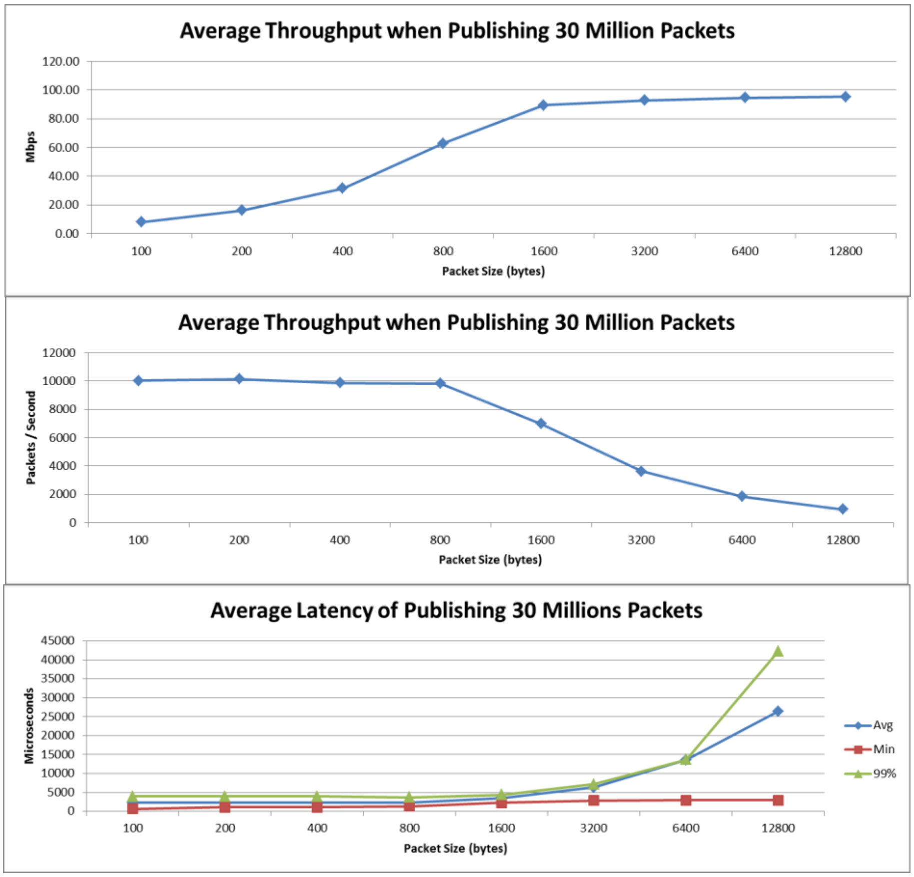
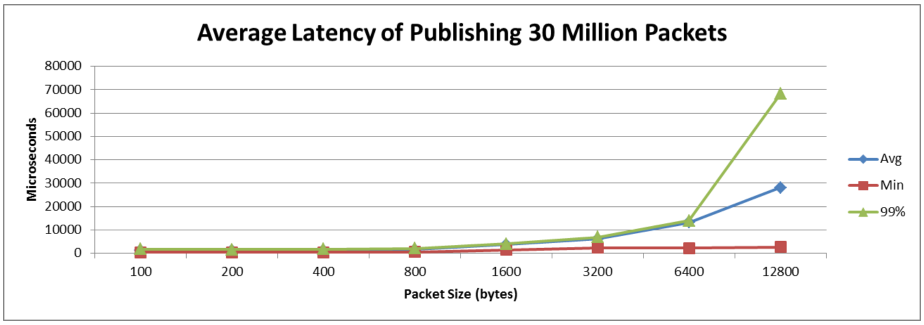


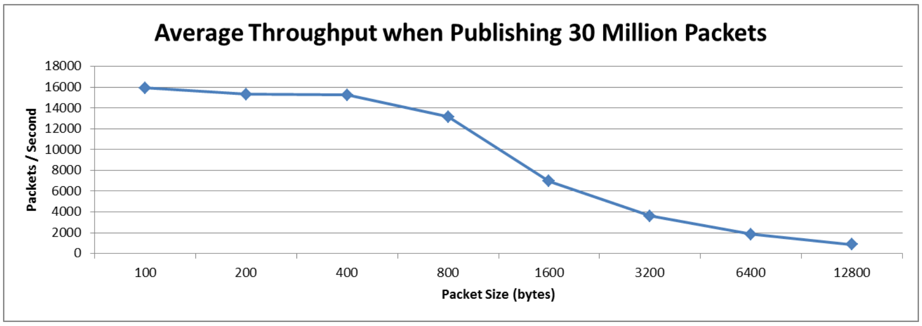
Figure 15 - Average Latency and Throughput Performance Cross 100Mbps Fast Ethernet Switch

The communication latency is consistently about 2ms with packet size up to 800 bytes. The communication latency starts to increase significantly when the packet size increases beyond 800 bytes and would reach to 26ms with packet size of 12,800 bytes. The reason for this increase in communication latency is obvious from the throughput graph where the middleware starts consuming the maximum bandwidth of the Ethernet communication switch of 100 Mbps with packet size of 1,600 bytes. Since the quality of service is set to reliable communication, the middleware starts blocking the packets and throttle the communication with maximum bandwidth available close to 100Mbps. This clearly demonstrates that the throughput is limited by the network capability and not by the CPU or real-time DDS middleware. Although the communication latency is very high with packet size of 12,800 bytes compared to that with packet size of 800 bytes, it is still within the required scan time resolution of 35ms.

### 6.3.5. *Impact Analysis of using Different Controller and I/O System Machines*

The next performance test is to demonstrate the impact of having two unequal laptops with different CPU speed. The same configuration is used in the previous test. In the first test, the controller application is hosted in the faster machine. However, in the second test, the controller application is hosted in the slower machine. For the first test, the measured average latency and throughput in terms of packets per second and Megabits per second for the first test are shown in Figure 16.





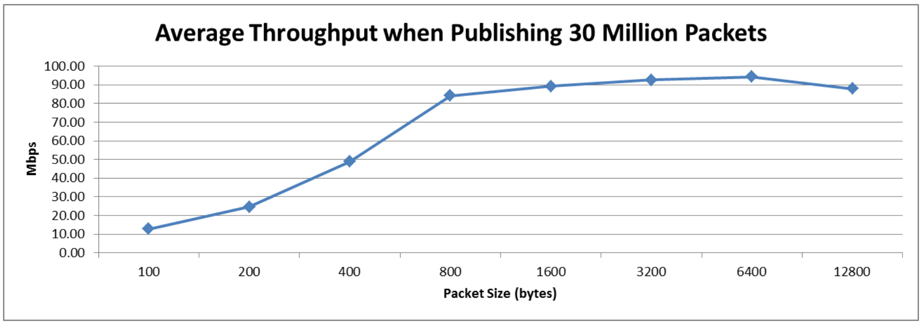
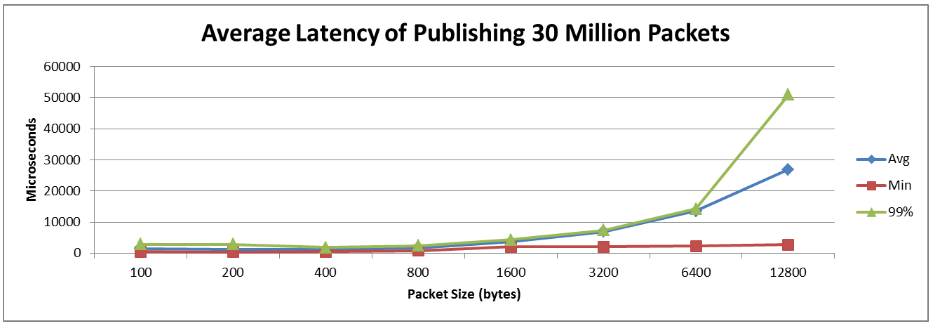
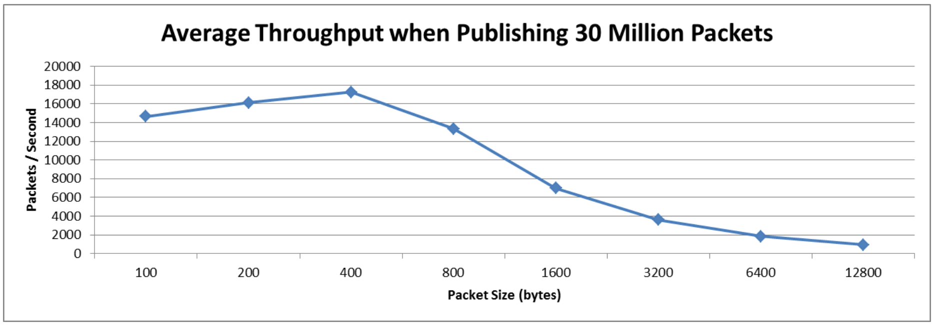


Figure 16 – Average Latency and Throughput Performance with Fast Controller

For the second test, the measured average communication latency and throughput in terms of packets per second and Megabits per second for the first test are shown in Figure 17. The performance is improved when the I/O system is hosted in a faster machine.





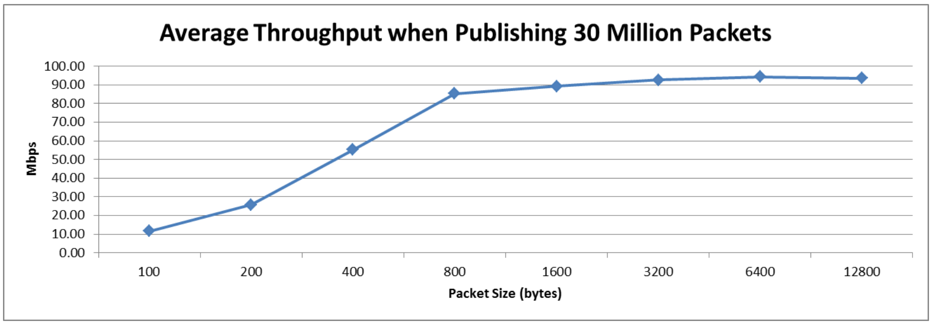


Figure 17 – Average Latency and Throughput Performance with Slow Controller

However, the performance has improved further when the fast machine is used to host the controller application rather than the I/O system. Because the real-time DDS middleware uses true peer-to-peer messaging with no centralized or message broker, server, or daemon processes, it does not impose any inherent limit on the aggregate messaging capacity. It is limited only by the network infrastructure. In all cases, for large systems with packet size beyond 1,600 bytes, it is more efficient to use higher network bandwidth capacity such as the 1 Gbps Ethernet switch.

## 7. Conclusion

In today's competitive production environment, there is a very high demand on addressing the obsolescence challenges of process automation systems utilized in oil and gas process industries. Extending the life cycle of these systems through incremental upgrades, migration paths, or partial components replacement would lessen the impact of obsolescence challenges to a limited extent and cannot resolve them. The essential strategy enabler in today's world for safe guarding against premature obsolescence challenges is the utilization of COTS, open source, and/or multi-supplier technologies in order to move from proprietary to standards-based automation solution. To achieve this strategy, this paper presents a new design of the automation controllers to allow upgrades or replacements of any obsolete components without the need to buy another automation controller altogether. The main concept of this approach is the physical and logical decoupling of the I/O systems from the current proprietary monolithic controllers. However, this change requires real-time reliable and fault tolerant data-centric middleware that provides seamless cross-vendor interoperability. Detailed performance analysis was conducted to evaluate the viability of utilizing the real-time publish/subscribe DDS middleware as a core communication link between the controller and the I/O systems.

The performance result of the average communication latency between the controller and the I/O system in all tests is very well below the required scan time resolution while varying the controller size from 100 bytes equivalent to a PLC with 400 digital I/O and 50 analog I/O, to a controller size of 12,800 bytes equivalent to a PLC with 80,000 digital I/O and 2,800 analog I/O. Because the real-time publish/subscribe DDS middleware uses true peer-to-peer messaging with no centralized or message broker, server or daemon processes, the performance tests showed that it does not impose any inherent limit on the aggregate messaging capacity. The main limiting factor is the selected network infrastructure.

The following are the advantages of the multi-supplier COTS-based heterogeneous automation controller:

- It is a cost effective evergreen solution because it is based on field proven interoperable and standard multi-supplier COTS software and hardware components resulting in optimum capital investment for the total cost of ownership throughout the life span of the processing facility form hurdle to grave.
- It is a high-performance controller because of the decoupling of the I/O systems from the controller. The I/O scan update is processed during the control application scan cycle.

- It is based on a distributed architecture where I/O modules, CPU and control application are not interwoven. Changing process I/O signals does not have any impact on the control application.

## Acknowledgments

The authors acknowledge King Fahd University of Petroleum & Minerals for all support.

## References

1. ABB, *IndustrialIT System 800xA System Architecture,* White Paper, (2005)
2. ARC Advisory Group, www.arcweb.com/market-studies
3. ARC Advisory Group, *Honeywell Experion PKS R300*, White Paper, (2005)
4. ARC Advisory Group, *Emerson's Built for Purpose Approach to Commercial-Off-The-Shelf Technology,* White Paper, (2010)
5. C. Mascolo, S. Haile, L. Lymberopoulos, G. Picco, P. Costa, G. Blair, P. Okanda, T. Sivaharan, W. Fritsche, M. Karl, M. Ronai, K. Fodor, and A. Boulis, *Survey of Middleware for Networked Embedded Systems*, Sixth Framework Program Priority 2, Information Society technologies, IST-004536-RUNES - D5.1 1.0, pp. 1-83, (2005)
6. C. Pereira and L. Arro, *Distributed Real-Time Embedded Systems: Recent Advances, Future Trends and Their Impact on Manufacturing Automation*, Annual Reviews in Control, **31** (2007), pp. 81-92
7. D. Coughanowr and S. LeBlanc, *Process Systems Analysis and Control* (McGraw-Hill, 2008)
8. F. Chen, and T. Repantis, *Coordinated Media Streaming and Transcoding in Peer-To-Peer Systems*, 19th International Parallel and Distributed Processing Symposium, (2005), pp. 56b
9. G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications* (Springer, 2011)
10. G. Coulouris, *Distributed Systems: Concepts and Design* (Addison-Wesley, 2011)
11. G. McKim, M. Yeager, and C. Weirich, *DCS Upgrades for Nuclear Power Plants: Saving Money and Reducing Risk through Virtual-Simulation Control System Checkout*, Foxbro SimSci-Esscor, White Paper, (2011)
12. G. McMillan and D. Considine, *Process/Industrial Instruments and Controls Handbook* (McGraw-Hill, 1999)
13. G. Pardo-Castellote, *Data-Centric Programming Best Practices: Using DDS to Integrate Real-World Systems,* Real-Time Innovations, Inc., (2010), pp. 1-18
14. G. Pardo-Castellote, *OMG Data-Distribution Service: Architectural Overview*, Real-Time Innovations, Inc., (2005), pp. 1-7
15. K. An, A. Gokhale, D. Schmidt, S. Tambe, P. Pazandak and G. Pardo-Castellote, *Content-based Filtering Discovery Protocol (CFDP): Scalable and Efficient OMG DDS Discovery Protocol*, DEBS '14, (2014), pp. 1-12
16. K. Ramamritham and S. Son, *Real-time databases and data services,* Real-Time Systems, **28/2** (2004), pp. 179-215
17. K. Sawada, *Achieving an Innovative Unified Operation Environment Using the Unified Gateway Station (UGS),* Yokogawa Technical Report, **54/2** (2011)
18. L. Zhai, L. Guo, X. Cui and S. Li, *Research on Real-time Publish/Subscribe System supported by Data-Integration*, Journal of Software, **6/6** (2011) pp. 1133-1139
19. L. Zou, Z. Wang, H. Dong, Y. Liu and H. Gao, *Time- and Event-Driven Communication Process for Networked Control Systems: A Survey*, Hindawi Publishing Corporation, Article ID 261738 (2014), pp. 1-10

20. M. Anand, S. Sarkar and S. Rajendra, *Application of Distributed Control System in Automation of Process Industries*, International Journal of Emerging Technology and Advanced Engineering, **2/6** (2012), pp. 377-383
21. M. Mahmoud and Y. Xia, *Analysis and Synthesis of Fault-Tolerant Control Systems* (Wiley, 2014)
22. M. Vernak and T. Shope, *Justification for DCS Migration*, Rockwell Automation White Paper, (2014)
23. N. Medvidovic,*The Role of Middleware in Architecture-Based Software Development*, International Journal of Software Engineering and Knowledge Engineering **13/4** (2003), pp. 367-393
24. Object Management Group, *Data Distribution Service for Real-Time Systems Specification*, Version 1.2, (2007)
25. S. Sait and G. Hashim, *Novel Design of Collaborative Automation Platform for Optimum Process Control Environment*, Journal of Circuits, Systems, and Computers, **25/6** (2016)
26. W. Bolton, *Programmable Logic Controllers* (Newnes, 2005)
27. W. Heinzelman, A. Murphy, H. Carvalho and M. Perillo, *Middleware to Support Sensor Network Applications*, IEEE Network Magazine, **18** (2004)
28. W. Levine, *The Control Handbook* (CRC Press, 2010)

**Sadiq M. Sait** obtained a Bachelor's degree in Electronics from Bangalore University in 1981, and Master's and PhD degrees in Electrical Engineering from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in 1983 & 1987 respectively. Since 1987 he has been working at the Department of Computer Engineering where he is now a Professor. In 1981

Sait received the best Electronic Engineer award from the Indian Institute of Electrical Engineers, Bangalore (where he was born). In 1990, 1994 & 1999 he was awarded the 'Distinguished Researcher Award' by KFUPM. In 1988, 1989, 1990, 1995 & 2000 he was nominated by the Computer Engineering Department for the 'Best Teacher Award' which he received in 1995, and 2000. Sait has authored over 200research papers, contributed chapters to technical books, and lectured in over 25 countries. Sadiq M. Sait is the principle author of the books (1) VLSI PHYSICAL DESIGN AUTOMATION: Theory & Practice, published by McGraw-Hill Book Co., Europe, (and also co-published by IEEE Press), January 1995, and (2) ITERATIVE COMPUTER ALGORITHMS with APPLICATIONS in ENGINEERING (Solving Combinatorial Optimization Problems): published by IEEE Computer Society Press, California, USA, 1999. He was the Head of Computer Engineering Department, KFUPM from January 2001 – December 2004, Director of Information Technology and CIO of KFUPM between 2005 and 2011, and now is the Director of the Center for Communications and IT Research at the Research Institute of KFUPM.

**Ghalib A. Al-Hashim** obtained a Bachelor's degree in Computer Engineering from University of Arizona, Tucson, USA in 1988, Master degree in Computer Engineering from KFUPM in 1998, Master degree in Business Administration from KFUPM in 2008, and PhD degree in Computer Science and Engineering from KFUPM in 2016. Ghalib is a certified automation professional by International Automation Society. He is also a certified functional safety professional by TUV and CFSE Governance Board. He has been working in Saudi Aramco for more than 32 years specialized on process computer systems. From 2007 until 2014, he was the Corporate Process Automation Work Director responsible of the process automation business and capital plan covering all Saudi Aramco facilities including upstream, gas processing, refining, petrochemical, oil & gas pipelines, tank farm, bulk plants and products distribution.