Journal of Circuits, Systems, and Computers

World Scientific
www.worldscientific.com

NOVEL DESIGN OF COLLABORATIVE AUTOMATION PLATFORM USING REAL-TIME DATA DISTRIBUTION SERVICE MIDDLEWARE FOR AN OPTIMUM PROCESS CONTROL ENVIRONMENT

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 (16 June 2015)
Revised (06 December 2015)
Accepted (08 December 2015)

Refining and petrochemical processing facilities utilize various process control applications to raise productivity and enhance plant operation. Client-server communication model is used for integrating these highly interacting applications across multiple network layers utilized in distributed control systems.

This paper presents an optimum process control environment by merging sequential and regulatory control, advanced regulatory control, multivariable control, unit-based process control, and plant-wide advanced process control into a single collaborative automation platform to ensure optimum operation of processing equipment for achieving maximum yield of all manufacturing facilities.

The main control module is replaced by a standard real-time server. The input/output racks are physically and logically decoupled from the controller by converting them into distributed autonomous process interface systems. Real-time data distribution service middleware is used for providing seamless cross-vendor interoperable communication among all process control applications and distributed autonomous process interface systems.

Detailed performance analysis was conducted to evaluate the average communication latency and aggregate messaging capacity among process control applications and distributed autonomous process interface systems. The overall performance results confirm the viability of the new proposal as the basis for designing an optimal collaborative automation platform to handle all process control applications. It also does not impose any inherent limit on the aggregate data messaging capacity, making it suitable for scalable automation platforms.

Keywords: Controller; Process Control; Real-Time; Data Distribution Service Middleware, Collaborative Automation.

1. Introduction

For the past forty years, the development of process automation systems including distributed control system (DCS) has been evolving to raise throughput and yield,

enhance plant operation and energy efficiency, and maintain consistent quality. Although DCS technology has advanced rapidly since the mid-1980s, the latest systems still follow the traditional hierarchical structure or network layers: field control network, plant control network, and plant information network. The main function of process control applications at the lowest network layer is to ensure a particular process doesn't vary by more than an allowed amount. It monitors the operation of each part of the process, identifies unwanted changes and initiates any necessary corrective actions. Further up the hierarchical structure, the control system at the middle network layer controls the overall production process and makes sure it continues to operate efficiently. Finally at the plant network layer, the advanced control applications ensure optimum performance of processing units and improve the speed and accuracy of controlling a process, through reduced variability and increased predictability to provide the operators with an opportunity to optimize the process by moving its operational setting closer to its constraints, where typically the performance is higher [27].

The interaction among the heterogeneous process control applications across all network layers are conventionally implemented using client-server communication model. This communication model works very well for the conventional system architecture where there is a centralized server in each network layer. At the field control network layer, the process data is centralized within each associated controller. At the plant control network layer, the process data is centralized within a data server providing real-time process data to all proprietary nodes from the same system vendor at this layer. At the plant network layer, the process data is centralized within an object linking and embedding for process control (OPC) data server providing near real-time plant data to other nodes from different manufacturers at this layer. Unfortunately, this model precludes deterministic communications and is not effective for exploiting the processing facilities to achieve maximum yield since the information is being generated at multiple nodes and the client does not know when new information is available [6].

The objective of this paper is to present a solution for addressing the limitation in client-server communication model deployed in refining and petrochemical industries for integrating highly interacting process control applications across multiple network layers utilized in distributed control systems. 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 among all process control applications in order to realize components' interoperability. This concept will be an enabler for designing an optimum process control environment through an efficient and effective integration of sequential and regulatory control, advanced regulatory control, multivariable control, unit-based process control, and plant-wide advanced process control in a single collaborative automation platform to ensure optimum operation of processing equipment for achieving maximum yield of all manufacturing facilities.

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 refining and petrochemical production environment coupled with strict government laws and environmental regulations, there is a very high demand on process industries to maximize valuable products while maintaining high quality and minimizing required energy consumption for survival and sustainability in the business. This challenging requirement mandates the utilization of latest agile and rigorous process control technology to increase productivity, improve quality, and minimize cost [10].

Figure 1 illustrates a typical distributed control system architecture deployed in refining and petrochemical industries.

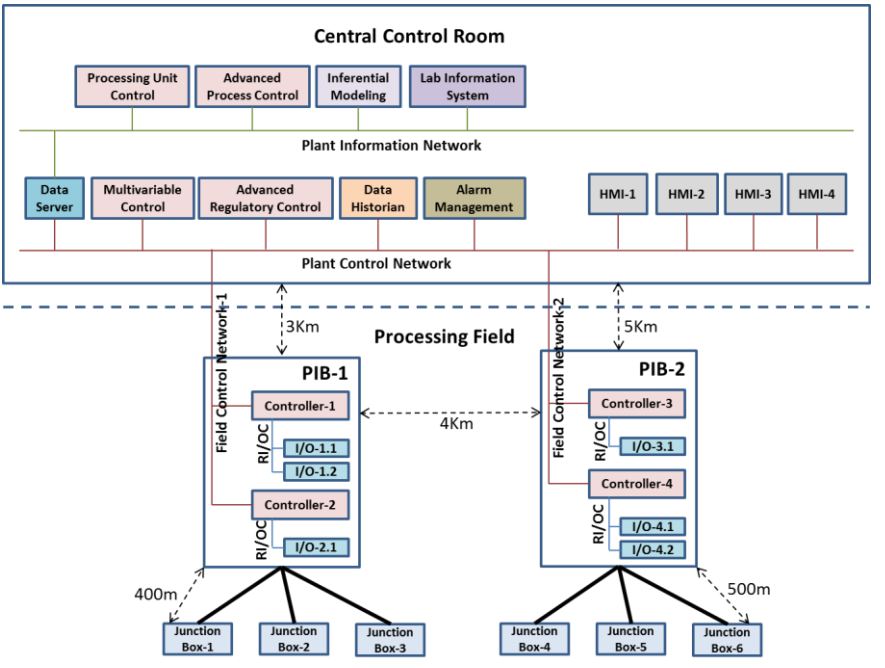

Figure 1 -Typical Distributed Control System Architecture

There are three different network layers: field control network layer, plant control network layer, and plant information network layer. The architecture is based on multiple proprietary or Ethernet-based local area field control networks, called control segments,

extended from the central control room (CCR) to the process interface buildings (PIB)-1 and PIB-2, located close to the processing facilities. Each control segment connects two controllers responsible for interfacing to field measurement and control devices. The control segments are interfaced to a proprietary or Ethernet-based local area plant control network, located in the CCR, that connects four human machine interface (HMI) consoles, alarm management, data historian, data server, advanced regulatory control, and multivariable control. The data server is also connected to a proprietary or Ethernet-based local area plant information network to provide near real-time process data to all nodes in this network layer including processing unit control, advanced process control, inferential modeling for predicting unmeasured process properties, and lab information system for calibrating the inferential models.

The function of the controller is to perform the basic control strategies including proportional, integral and derivative (PID) continuous control loops and discrete sequence control in order to facilitate the basic operation, control and automation requirements. The controllers are distributed in this layer horizontally through control segments. Each controller is connected to its associated I/O racks using a proprietary remote I/O communication (RI/OC) link. The 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 components but are all interwoven. The plant control network backbone above the control segments includes the multivariable control to ensure minimum control loops interactions for achieving optimum performance of processing equipment, and advanced regulatory control to provide feed forward, adaptive gain, and fuzzy logic control [12].

Moving vertically one layer above the plant control network is the unit and advanced process control. The unit process control is to ensure optimum performance of processing units. The advanced process control is to improve the speed and accuracy of controlling a process, through reduced variability and increased predictability to provide the operators with an opportunity to optimize the process by moving its operational setting closer to its constraints, where typically the performance is higher. To achieve optimum advanced process control solution, inferential models are used to provide near real-time estimates of product qualities otherwise available only through infrequent online or laboratory analysis. Inferential models must be calibrated using lab or on-line analyzer measurements to maintain their accuracy. The lab information system is used for this purpose [7].

The process control technology ranges in complexity from basic discrete control up to advanced process control as shown in Figure 2.

Discrete sequential control and continuous regulatory control including local control loops, with only one output being controlled by a single manipulated variable, are implemented in the controllers at the field control layer. The main function of the

regulatory control is to correct the controlled process after detecting any predictable upsets [25].

Multivariable and advanced regulatory controls including adaptive control, feed forward and fuzzy logic control schemes require complex computation power that normally does not exist in conventional controllers. Therefore, they are implemented at the plant control layer using proprietary application modules. The main function of the multivariable controller is to decouple highly interacting control loops and make them operate successfully in tandem. The main function of the advanced regulatory control is to correct the controlled process before any predictable upsets happen and/or after detecting any unpredictable upsets [20].

Process unit and advanced process controls including unit performance governor and plant optimization require intensive computation power, not available in conventional controllers and application modules. Therefore, they are implemented using high-performance servers at the plant information layer. The main function of these control applications is to ensure optimum performance of processing units and to optimize the process by moving its operational setting closer to its constraints [27].

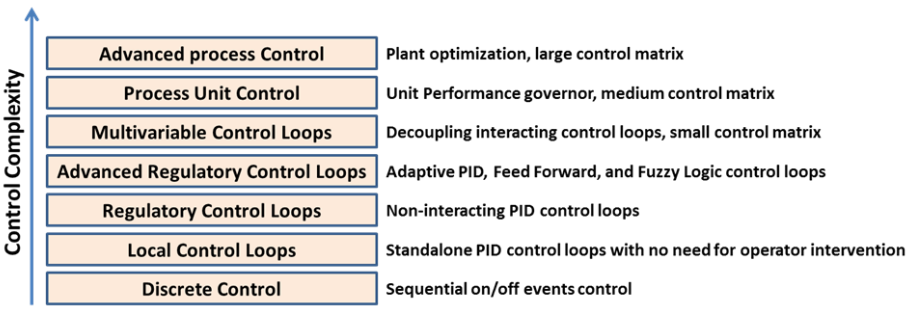

Figure 2 – Complexity Range of Process Control Applications

3. Related Work

Modern control engineering is a relatively new field of study that gained significant attention during the 20th century with the advancement of technology.

3.1. *Distributed Control System Evolution*

In the 1980s, network-centric automation systems evolved resulting in distributed control system architectures. It is believed that higher performance can be achieved if greater amounts of data are shared throughout the enterprise using open systems. Foxboro was the first DCS supplier to adopt UNIX and its companion Ethernet networking technologies along with their own proprietary protocol layers. This made it possible to implement the first instances of object management and global data access technology [1,3,11].

The drive towards openness in the 1980s gained momentum through the 1990s with the increased adoption of commercial off-the-shelf (COTS) components and IT standards resulting in application-centric automation systems. Utilizing COTS components not only resulted in lower manufacturing costs for the supplier, but also decreased prices steadily for the end users, who were also becoming increasingly vocal over what they perceived to be excessively high proprietary hardware costs. The biggest transition undertaken during this era was the development of OPC technology and the move from the UNIX operating system to the Windows environment for everything above the controllers. Standard computer components from manufacturers such as Intel and Motorola made it cost prohibitive for DCS suppliers to continue making their own workstations and networking hardware. Hence, the primary business of DCS suppliers, for years, had been the supply of large amounts of proprietary hardware, particularly controllers and associated I/O racks. The drive towards collaboration among applications in the 1990s gained momentum through the 21st century with the increased adoption of object-centric protocol for the interface among web applications using client-server communication models [2,4,22].

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

Middleware is a collection of technologies and services that enables 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 by Medvidovic [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 becomes available [26].

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 focus is the interface between the applications. [14,15].

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 require specific features including efficiency, determinism, flexibility delivery bandwidth, and fault-tolerant operation [13,19]. Below we briefly discuss each of these features.

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. The computing power of the main control module is limited and not suitable for computing intensive control strategies. As a result, proprietary application

modules are used for implementing advanced regulatory and multivariable control strategies at the plant control network layer. The plant control layer is proprietary and cannot accommodate any third party applications. Hence, the standard rigorous solutions for advanced process control have been implemented at the plant network layer utilizing near real-time process data provided by the OPC data server.

The current client-server communication model utilized for integrating highly interactive heterogeneous process control applications across the vertical and horizontal network layers is not effective for exploiting the processing facilities to achieve maximum yield. This is because the information is being generated at multiple nodes and the client-server model is inefficient and precludes deterministic communications, since the client does not know when new information is available.

To address the above limitation on client-server models for real-time systems to achieve optimum process control environment, two features must be accomplished; moving away from the multiple-network-layer architecture and from the data centralization at the controllers and data servers.

5. Proposed Solution

In this paper authors design a novel collaborative automation platform to solve the current monolithic controllers' architecture problem by physically and logically decoupling the I/O racks from the main control module, converting them into distributed autonomous process interface systems, and relocating them to the field junction boxes. The main control module is replaced by a standard server-based controller running on a standard real-time operating system environment. This change is accomplished by employing real-time reliable and fault-tolerant data centric middleware that provides seamless cross-vendor interoperability. The OMG DDS is the first open international middleware standard, suitable for addressing publish-subscribe communication for real-time data-critical applications and embedded systems.

The above modularity concept results in the development of a collaborative automation platform centered on distributed autonomous process interface systems and real-time DDS middleware to provide effective and efficient publish-subscribe communication among all process control applications as shown in Figure 3.

The following is a detailed example of the proposed control system architecture.

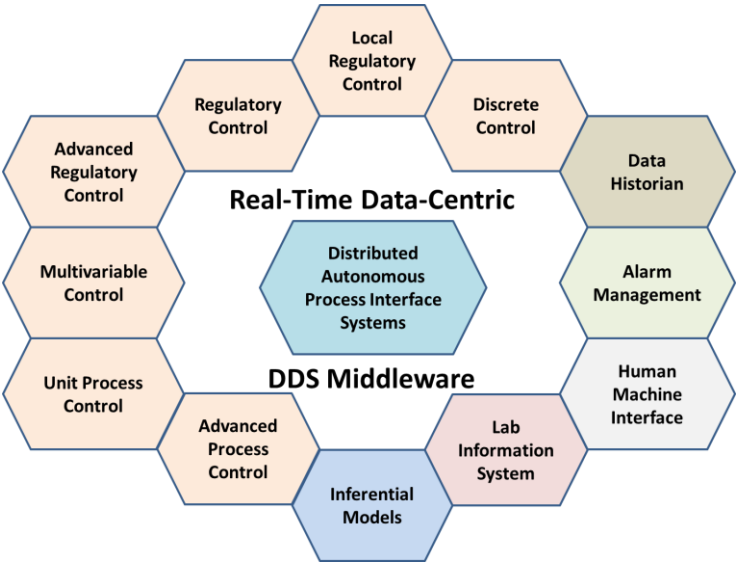

Figure 3 – Collaborative Automation Platform

5.1. *Proposed Control System*

The equivalent distributed control system architecture based on the proposed concept is shown in Figure 4. The new controller architecture consists of three main components empowered by real-time reliable and fault tolerant DDS middleware; control servers, I/O processing and inferential modeling servers, and distributed autonomous process interface systems. The control servers host software-based control applications ranging in control complexity from basic discrete control up to advanced process control as shown in Figure 3. As illustrated in Figure 4, all field instruments are hardwired to the field junction boxes close to the associated process equipment. There are two types of junction boxes; master junction box and smart junction box.

The smart junction box includes terminal strips to interface with the wiring cables connecting the electrical signals to the associated instruments and field devices and the autonomous process interface system with required I/O modules to condition and convert the electrical signals to digital data, and a DDS-enabled communication adapter with dual Ethernet communication ports. The smart junction box is located within 50 meters from the processing equipment requiring control.

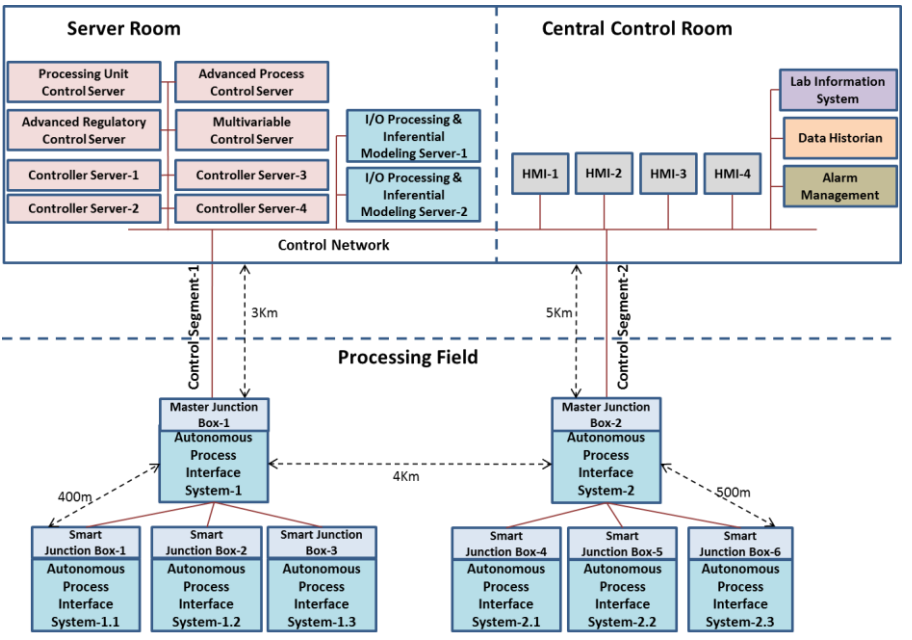

Figure 4 – Collaborative Automation System Architecture

The master junction box is similar to the smart junction box with an additional Ethernet switch as shown in Figure 5. Its location is selected to be as close as possible to the center of large number of junction boxes in order to minimize cabling cost.

The Ethernet switch uplink within each master junction box is a control segment extended from the local area control network in the CCR.

Each autonomous process interface system corresponds to a standalone DDS-enabled I/O system with required hardware for connecting various filed process measurement and control devices including pressure, flow, level, and temperature instruments, motion sensors, position sensors, and final device elements such as motor operated gate valves and control valves for regulating the process pressure, flow, level or temperature.

There are three types of autonomous process interface systems independent of the types of controllers' servers and their manufacturers: COTS I/O system, proprietary I/O system, and I/O bus networks. COTS I/O system type is the primary component of the proposed solution for the future automation applications.

The COTS I/O systems are currently used mainly for data acquisition and rarely for supervisory control through the web. There are three types of physical configurations, compact, fixed type modules, and mixed type modules. The compact I/O system comes with mixed types of I/O and fixed number of I/O channels. The fixed type modular I/O system comes with specific types of I/O and fixed number of modules with fixed number

of I/O channels. The mixed type modular I/O system comes with any types of I/O modules, but fixed number of modules with different number of I/O channels.

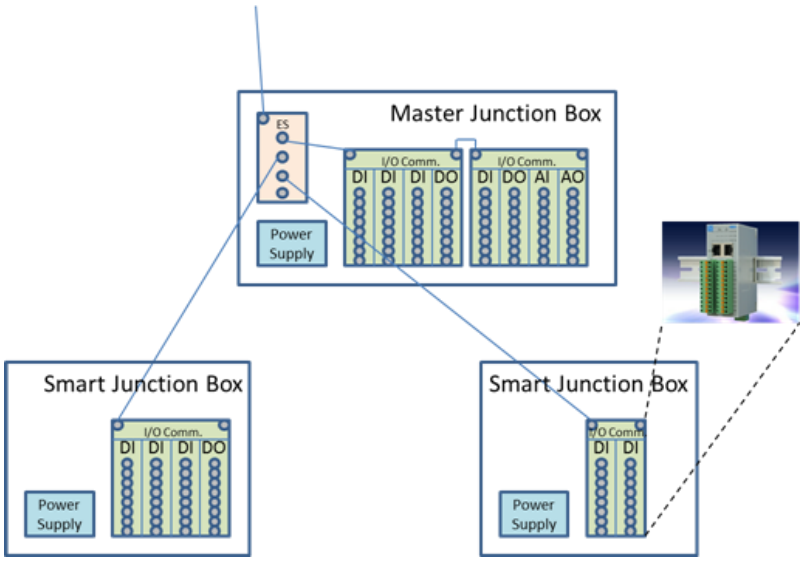

Figure 5 -Master and Smart Junction Boxes Relationship

The process interface systems are autonomous because they are self-contained and can run without the need for a controller specifically when it functions as a linking device for field bus networks. Each one of them can be associated by one or more of the software-based controllers as required. This is a significant advantage over the conventional control system architecture where a physical one-to-one relationship exists between the controller and its I/O racks. For example, if certain I/O signals are required for control strategies in two different controllers, these I/O signals are hardwired to both controllers through optical isolators.

Each operation area is assigned one I/O processing and inferential modeling server to process the I/O signals of the associated autonomous process interface systems and to infer unmeasured process properties required for the advanced process control applications. The function of the I/O processing server is to acquire the latest status changes of all hardwired input signals in the associated autonomous process interface systems, record sequence of events, generate applicable alarms, and publish the changed values to all subscribers requesting the latest updates including data historian and HMI consoles. In addition, it publishes the inferred process properties to all subscribers including multivariable control, processing unit control, and advanced process control applications through the middleware.

Each software-based controller hosts a single control application running continuously on a standard real-time operating system. The real-time control middleware layer is the heart

of the collaborative automation platform architecture and is based on the DDS middleware technology. This middleware is the most advanced and efficient standard-based technology for real-time data distribution and serves as a glue to connect the software-based control applications, I/O processing servers, distributed process interface systems, and the HMIs consoles. The communication relationships among publishers and subscribers are summarized in Table 1.

Table 1 – Publish/Subscribe Relationship of Topics

Publishers	Topics	Subscribers									
		1	2	3	4	5	6	7	8	9	10
1	Operation Commands		X								
	Override Commands				X						
2	Output Image				X						
3	Sequence of Events	X								X	
	Alarm	X									X
	Inferred Properties						X	X	X		
4	Input Image	X	X	X		X	X	X	X		
5	PID Set Points		X								
6	PID Set Points		X			X					
7	PID Set Points		X			X	X				
8	PID Set Points		X			X	X	X			
	Control Performance	X									
9	Historical Trends	X									
10	Alarm	X									

1	HMIs	6	Multivariable Control
2	Controllers	7	Unit Process Control
3	I/O Servers	8	Advanced Process Control
4	Process Interface Systems	9	Data Historian
5	Advanced Regulatory Control	10	Alarm Management

6. Performance Analysis

The proposed collaboration automation platform 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 6, 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.

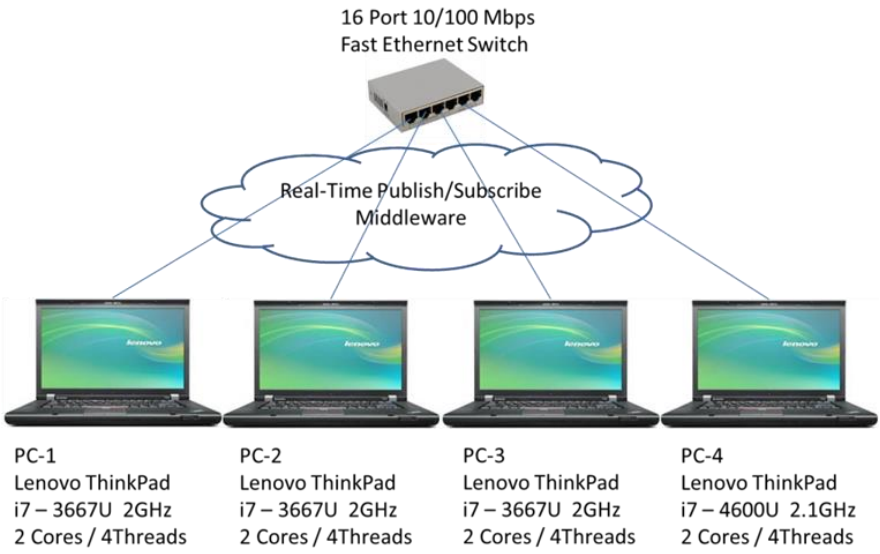

Figure 6 – COTS-Based Automation Controller

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 [24].

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 [24].

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 [24].

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 [24].

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 among the control applications and the autonomous process interface systems for safe and reliable operation of the processing facilities. The measuring performance criteria are the average latency and throughput.

6.4. *Data Simulation and Performance Test Measurement Methodologies*

The communication test between a publisher and a subscriber is as follows. The I/O system is the publishing side and the control application(s) 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.

6.4.1. *Communication Latency Measurement for a Single Subscriber*

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.

6.4.2. *Communication Latency Measurement for Multiple Subscribers*

The publisher uses the request for an echo exchange to measure the round-trip latency. There are two methods for measuring the communication latency when there is one publisher and many subscribers; unicast and multicast scenarios.

Unicast Scenario: The time stamp is logged by the publisher from the start of sending the data sample request consecutively to all subscribers until it receives the echo of the data sample back from the last subscriber as illustrated in Figure 7 for two subscribers. The communication latency between a publisher and the last subscriber, second subscriber in this case, is estimated by subtracting the one-way communication latency, as determined in a single publisher and a single subscriber case, from the roundtrip time to the last subscriber. In other words, the one-way latency is equal to T1 – T0 – one-way latency in one-to-one unicast case. The average communication latency between a publisher and the last subscriber is the average of the 30 thousand times of latency measurement during one test.

Multicast Scenario: The time stamp is logged by the publisher from the start of sending the data sample request to all subscribers until it receives the echo of the data sample back from the last subscriber as illustrated in Figure 8 for two subscribers. The communication latency between a publisher and the last subscriber, second subscriber in this case, is estimated by subtracting the one-way communication latency, as determined in a single publisher and a single subscriber case, from the roundtrip time to the last subscriber. In other words, the one-way latency is equal to T1 – T0 – one-way latency in one-to-one multicast case. The average communication latency between a publisher and the last subscriber is the average of the 30 thousand times of latency measurement during one test.

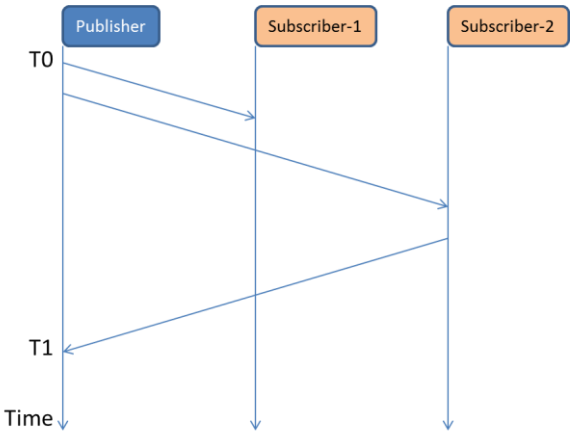

Figure 7 – One-to-Two Unicast Publish-Subscribe Communication Latency Measurement

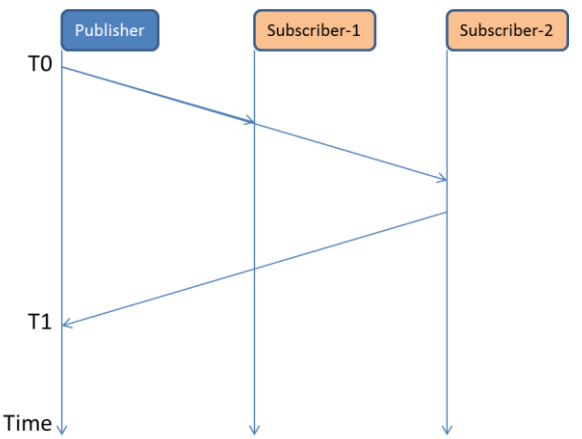

Figure 8 – One-to-Two Multicast Publish-Subscribe Communication Latency Measurement

6.5. *Baseline Performance Test Analysis (One Publisher to One Subscriber Scenarios)*

Each test scenario is repeated eight times with different data packet sizes 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 6 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 controller's 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 communication latency and throughput of one controller and one I/O system.

6.5.1. *Communication Latency and Jitter Analysis within One Laptop*

The measured average communication latency for the fourth laptop is shown in Figure 9. The performance result of the average communication latency between the controller and the I/O system is within 1ms, very well below the required scan time resolution while varying the controller size from 100 bytes equivalent to a controller with 400 digital I/O and 50 analog I/O, to a controller size of 12,800 bytes equivalent to a controller 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.

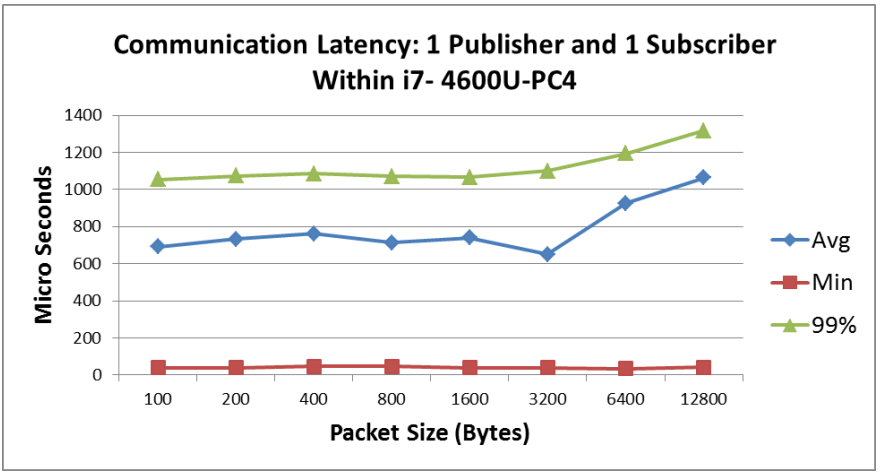

Figure 9 – Average Communication Latency between a Controller and an I/O system within one Laptop

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 red 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 for 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.

6.5.2. *Communication Throughput Analysis within One Laptop*

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, 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 fourth laptop measured in packets per second, represented by the blue series, and Megabits per second, represented by the red series, is shown in Figure 10.

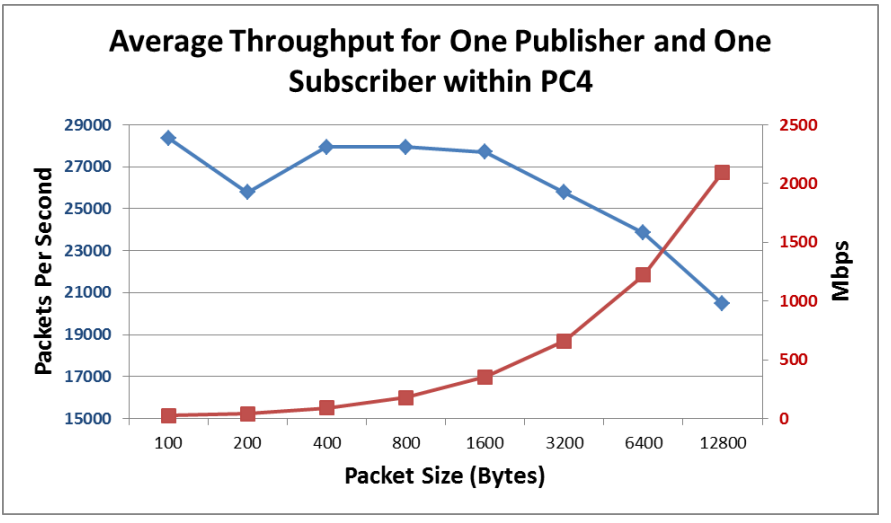

Figure 10 – Average Throughput for a Controller and an I/O system within one Laptop

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.

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

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 communication latency is shown in Figure 11. The measured average throughput bandwidth between one controller and one I/O system through fast Ethernet switch measured in packets per second, represented by the blue series, and Megabits per second, represented by the red series, is shown in Figure 12. The communication latency is consistently about 2ms with packet size up to 800 bytes, starts to increase significantly when the packet size increases beyond 800 bytes, and reaches 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.

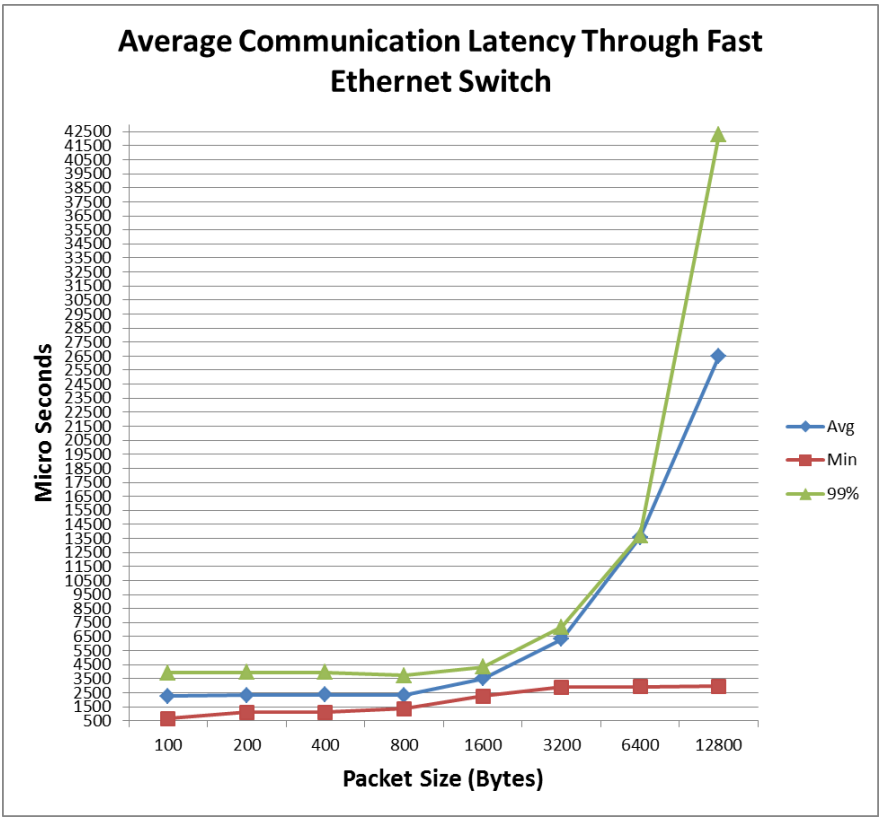

Figure 11 – Average Communication Latency through 100Mbps Fast Ethernet Switch

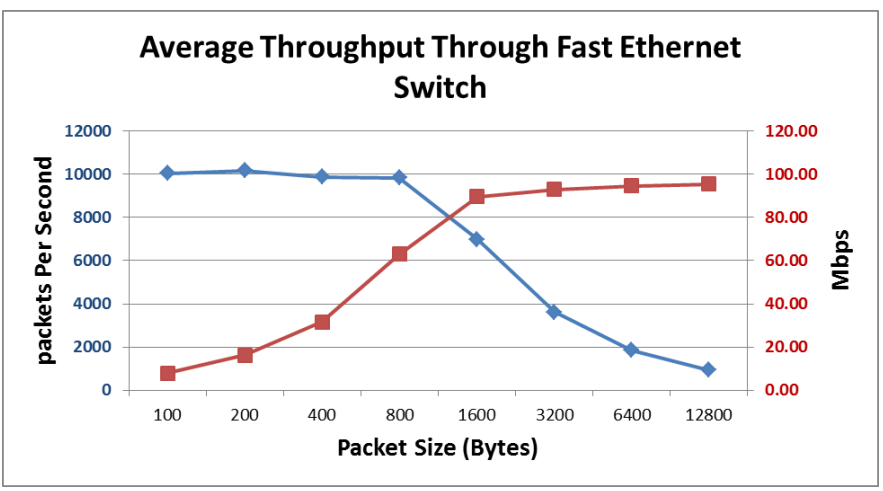

Figure 12 – Average Throughput Performance through 100Mbps Fast Ethernet Switch

6.5.4. *Communication Latency and Throughput Summary*

Figure 13 shows the overall communication latency for one publisher, I/O system, and one subscriber, control application, within each laptop and across 100Mbps Ethernet switch. There is no significant impact in terms of communication latency due to the middleware communication overhead within each laptop. Also, there is a minimum impact due to the Ethernet communication overhead on the performance if the packet size is within 800 bytes. However, with larger packet size beyond 800 bytes, the negative performance impact in communication latency is proportional to the transmission time of the packet through the Ethernet switch. The communication latency within 27ms is adequate for most process automation applications; however, if there is a special need for higher resolution scan time requiring less than 27ms latency, higher bandwidth communication network is recommended. The higher the communication bandwidth, the lower the communication latency that can approach 1ms as demonstrated for the case where both publisher and subscriber applications are within the same computing machine.

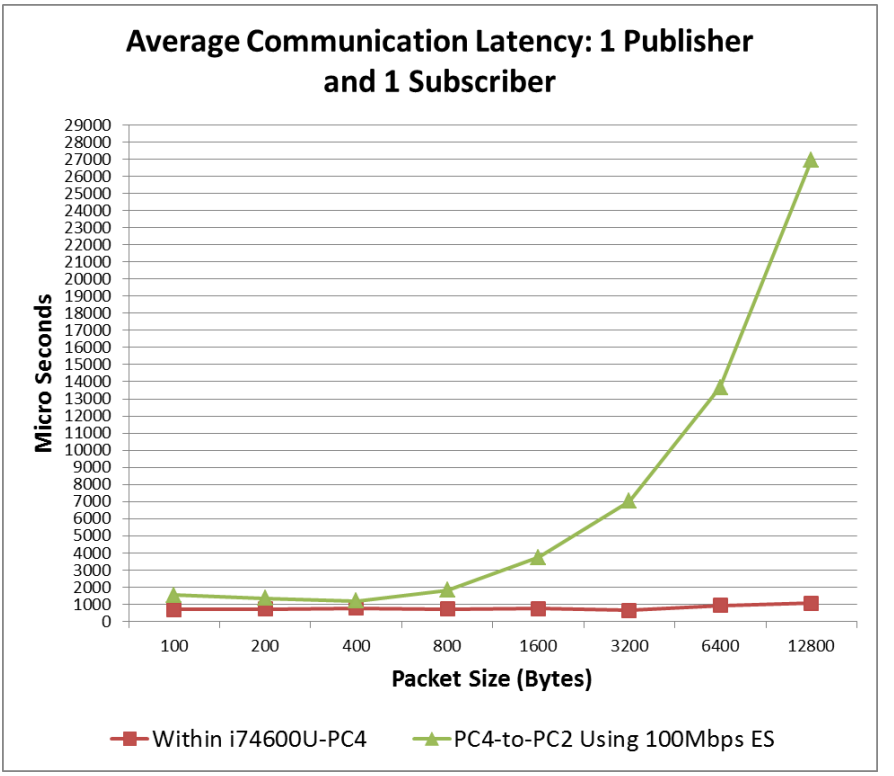

Figure 13 – Average Communication Latency Performance

Figure 14 shows the overall communication throughput for one publisher, I/O system, and one subscriber, control application, within each laptop and across 100Mbps Ethernet

switch. The blue series represent the throughput in terms of packets per second and the red series represent the throughput in terms of Megabits per second.

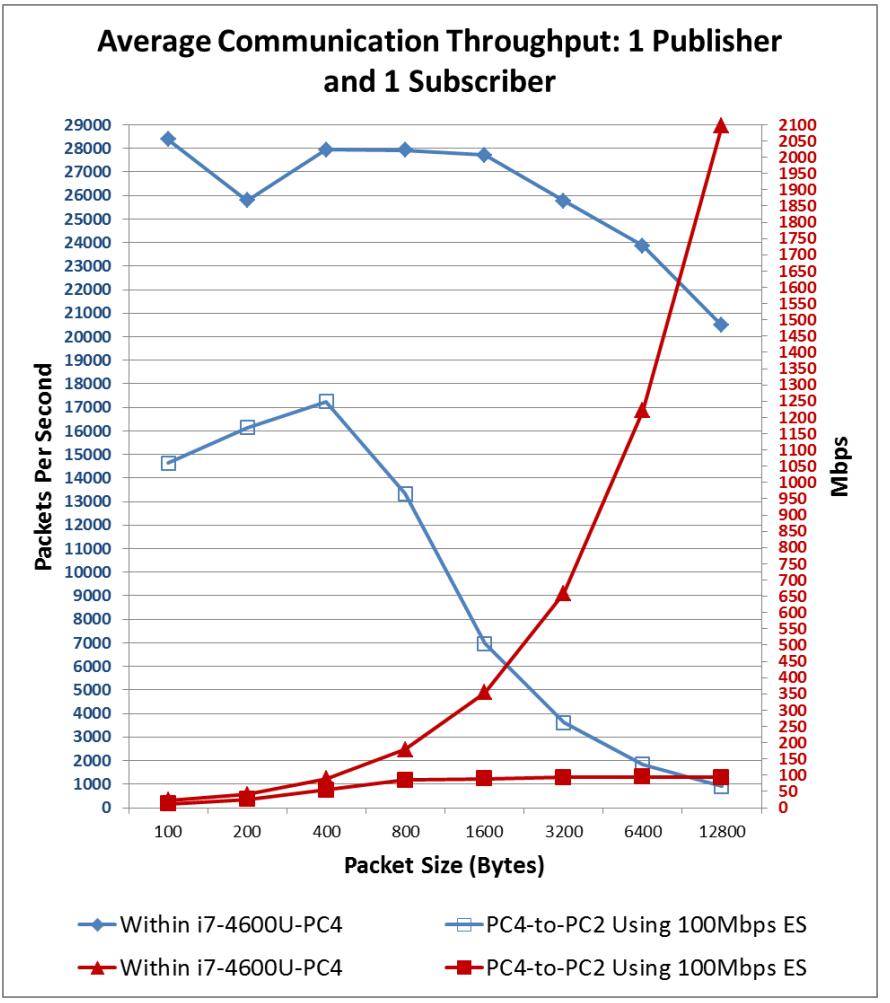

Figure 14 – Average Communication Throughput Performance

The optimum throughput of 28 thousand packets per second is achieved when the packet size equals 400 bytes where both publisher and subscriber applications are within the same computing machine. The communication throughput drops down to 20 thousands packets per second when using a packet size of 12800 bytes. However, it is about 18% higher than the optimum result achieved when the communication is crossing the 100Mbps Ethernet switch. Also, the decline slope for the communication throughput, when the packet size is more than 400 bytes, is sharper when the communication between the publisher and subscriber applications is performed through the Ethernet switch. The

communication throughput in terms of Megabits per second is calculated by multiplying the throughput rate in terms of packets per second times the size of the packet in bits. The middleware throttles the transmitted packets through the available fast Ethernet communication bandwidth of 100 Mbps.

6.6. *Performance Test Analysis (One Publisher to Multiple Subscribers Scenarios)*

For achieving optimum collaboration among process control applications, the publisher, the I/O system in this case, must provide reliably the right information at the right time to multiple subscribers, which are the associated control applications. Most of the subscribers require the process data information within seconds except for the discrete and regulatory control applications, within 100 milliseconds. To address the worst case scenario, we evaluate the middleware performance in terms of communication latency among all publisher and subscribers to be within 100 milliseconds. Each test scenario is repeated eight times with different data packet sizes 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 subscribers measure the throughput by counting the number of received data packets per second and the throughput rate of Megabits per second. The throughput is identical for all subscribers since the real-time middleware is designed with reliable communication quality of service. Therefore, the total throughput is the aggregate throughput at all subscribers. The performance test is to measure the actual communication latency and throughput of multiple control applications and one I/O system scenarios.

6.6.1. *Communication Latency and Throughput for One Publisher and Multiple Subscribers within One Laptop*

The set up for the next performance test configuration is to host one I/O system and multiple control applications, up to four subscribers, in one laptop. The measured average communication latency for four unicast communication scenarios with one, two, three, and four subscribers is shown in Figure 15. The performance result of the average communication latency between the multiple control applications and the I/O system is within 1ms, very well below the required scan time resolution while varying the controller size from 100 bytes to a controller size of 12,800 bytes. The data also shows that communication latency remains consistently low as message size increases for the four scenarios. It is noticed that the communication latency is doubled by increasing the number of subscribers by one. In other words, the communication latency is in the range of 100 micro seconds for one subscriber, 200 micro seconds for two subscribers, 400 micro seconds for three subscribers, and in the range of 800 micro seconds for four subscribers.

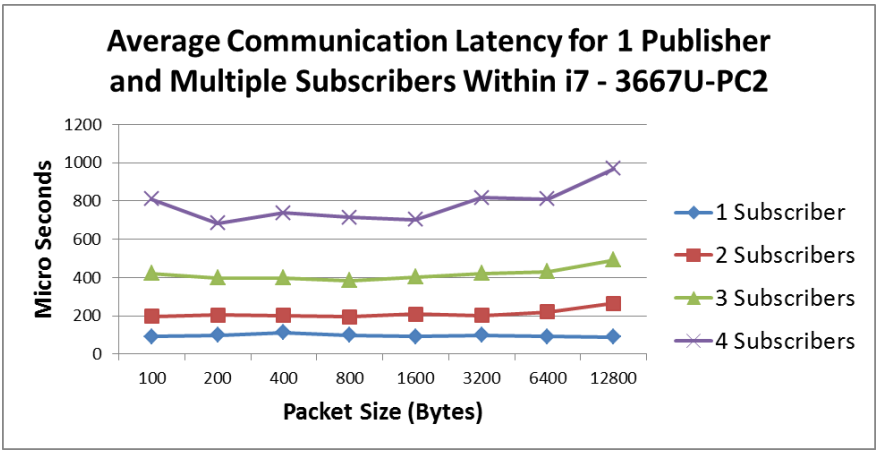

Figure 15 – Average Communication Latency Performance

The measured average throughput bandwidth per subscriber between multiple control applications and one I/O system for the fourth laptop measured in packets per second and Megabits per second is shown in Figure 16. The graph shows sustainable publish/subscribe throughput bandwidth. 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 control applications 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 collaborative automation platforms. The highest system throughput recorded in this experiment is 26828 packets per second, for the scenario with one publisher and four subscribers using 400- byte packet size. The lowest system throughput recorded in this experiment is 15128 packets per second, for the scenario with one publisher and one subscriber using 12800-byte packet size. The highest system throughput bandwidth recorded in this experiment is 2095 Mbps, for the scenario with one publisher and three subscribers using 12800-byte packet size. The lowest system throughput bandwidth recorded in this experiment is 16.4 Mbps, for the scenario with one publisher and one subscriber using 100-byte packet size.

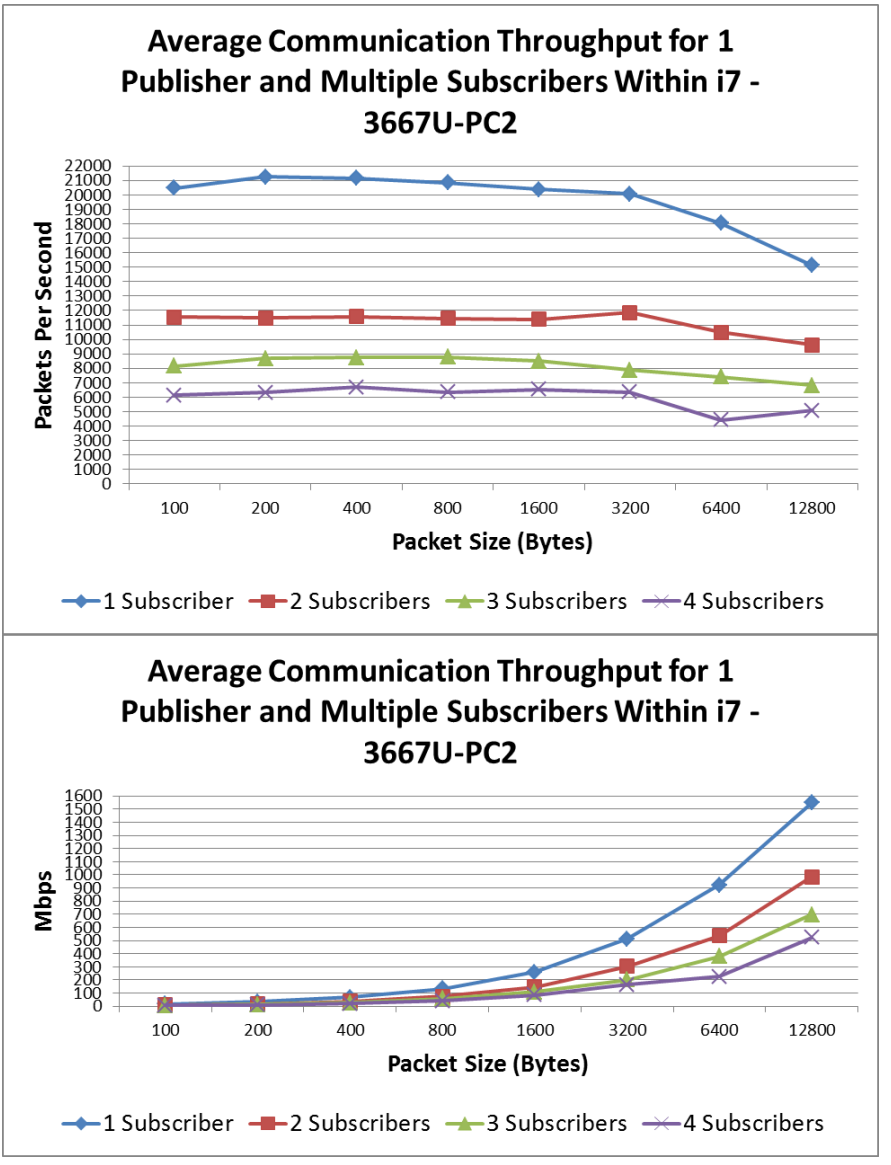

Figure 16 – Average Communication Throughput Performance per Subscriber

6.6.2. *Impact Analysis of using Ethernet Switch between One Publisher and Two Subscribers using Three Laptops*

The set up for the next performance test configuration is to host one I/O system in one laptop and two control applications in additional two laptops. The communication among the control applications 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 communication latency using unicast and multicast scenarios

compared to the baseline case, hosting the publisher and two subscribers within one laptop, is shown in Figure 17. The performance result of the average communication latency between two control applications and the I/O system for the three cases is within 2ms, very well below the required scan time resolution while varying the controller size from 100 bytes to a controller size of 400 bytes. As the packet size increases beyond 400 bytes, the average communication latency increases significantly using unicast communication with a packet size of 12800 bytes up to 50ms. On the other hand, there is a moderate proportional increase in the average communication latency using multicast communication to the size of the packet and reaches up to 28ms with a packet size of 12800 bytes. Therefore, multicast communication mode is the best method for communicating between a publisher and multiple subscribers in a collaborative automation platform.

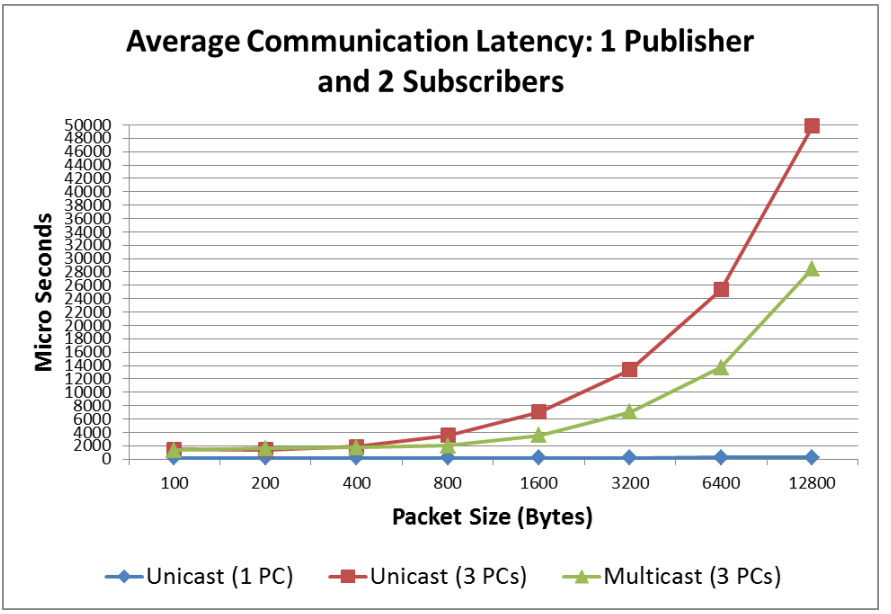

Figure 17 – Average Communication Latency Performance

The measured average communication throughput per subscriber using unicast and multicast scenarios compared to the baseline case, hosting the publisher and two subscribers within one laptop, is shown in Figure 18. The graph shows sustainable publish/subscribe throughput bandwidth between both control applications and the I/O system for the three cases in terms of packets per second while varying the controller size from 100 bytes to a controller size of 400 bytes. Obviously, there is a significant drop in the throughput per subscriber in terms of number of packets per second while increasing the size of the controller beyond 800 bytes due to the increase in transmission time of each packet and the requirement to throttle the communication with maximum bandwidth available close to 100Mbps.

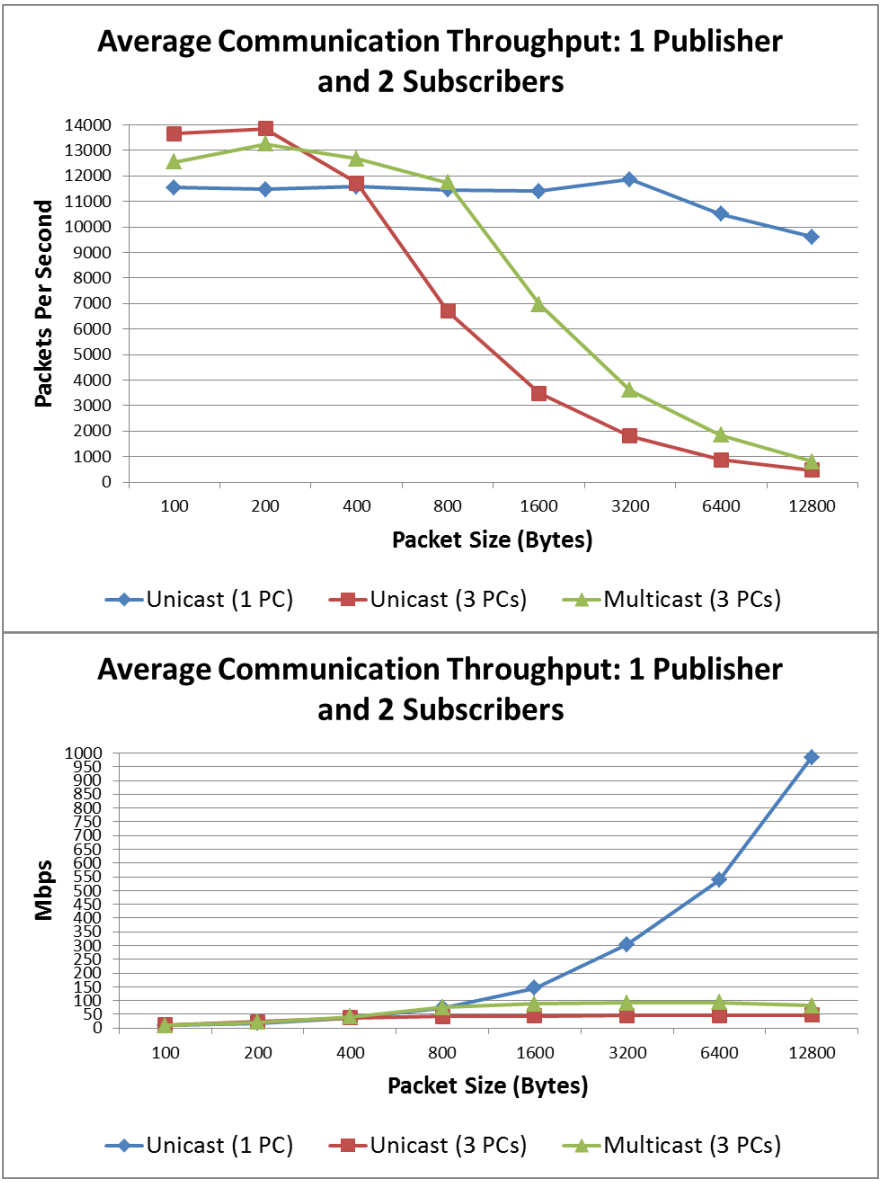

Figure 18 – Average Communication Throughput Performance per Subscriber

However, the multicast communication scenario shows better performance compared to the unicast communication in terms of packets per second. The multicast communication is best suitable for the real-time DDS middleware to cope with the huge increase in data loading and fully utilized available network communication bandwidth between the control applications 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

collaborative automation platforms. It is recommended to sustain the throughput for large collaborative automation platform beyond 400-byte packet size to use higher network bandwidth capacity such as the 1 Gbps Ethernet switch. This will restore the communication throughput performance close to the baseline case as shown in Figure 18 where the baseline throughput is less than 1Gbps for the largest controller using a packet size of 12800 bytes.

6.6.3. *Impact Analysis of using Ethernet Switch between One Publisher and Multiple Subscribers using Multicast Communication among Three Laptops*

The set up for the next performance test configuration is to host one I/O system in one laptop and multiple control applications evenly distributed in additional two laptops. The communication among the control applications 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 communication latency using multicast communication scenarios is shown in Figure 19. The scenarios include two subscribers, four subscribers, six subscribers, eight subscribers, ten subscribers and twelve subscribers corresponding to the collaborative applications illustrated in Figure 3.

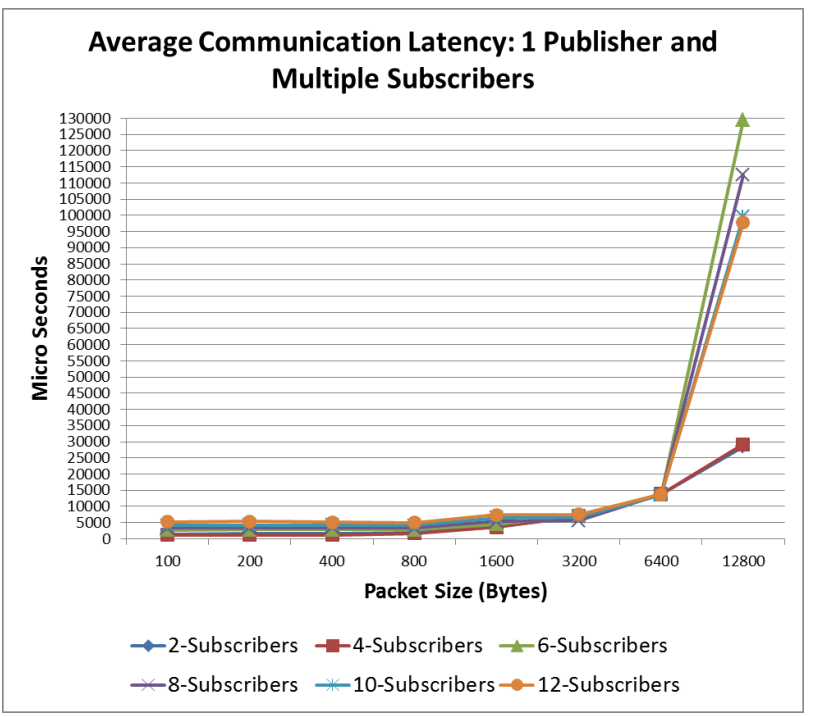

Figure 19 – Average Communication Latency Performance

The performance result of the average communication latency between multiple control applications and the I/O system for the six cases is within 15ms, very well below the required scan time resolution while varying the controller size from 100 bytes to a controller size of 6400 bytes. As the packet size increases beyond 6400 bytes, the average communication latency increases significantly for the cases of six, eight, ten and twelve subscribers with a packet size of 12800 bytes up to 128ms. This is due to the bandwidth limitation within the 100Mbps Ethernet switch. However, for the cases of two and four subscribers, the average communication latency is within 30ms and meeting the required scan time resolution. To reduce communication latency for more than four control applications below 35ms, the control network is required to be upgraded to a 1Gbps Ethernet infrastructure.

The measured average communication throughput in packets per second is shown in Figure 20. The graphs show sustainable publish/subscribe throughput bandwidth between the control applications and the I/O system for the six cases in terms of packets per second while varying the controller size from 100 bytes to a controller size of 800 bytes. Obviously, there is a significant drop in the throughput per subscriber in terms of number of packets per second while increasing the size of the controller beyond 800 bytes due to the increase in transmission time of each packet and the requirement to throttle the communication with maximum bandwidth available close to 100Mbps.

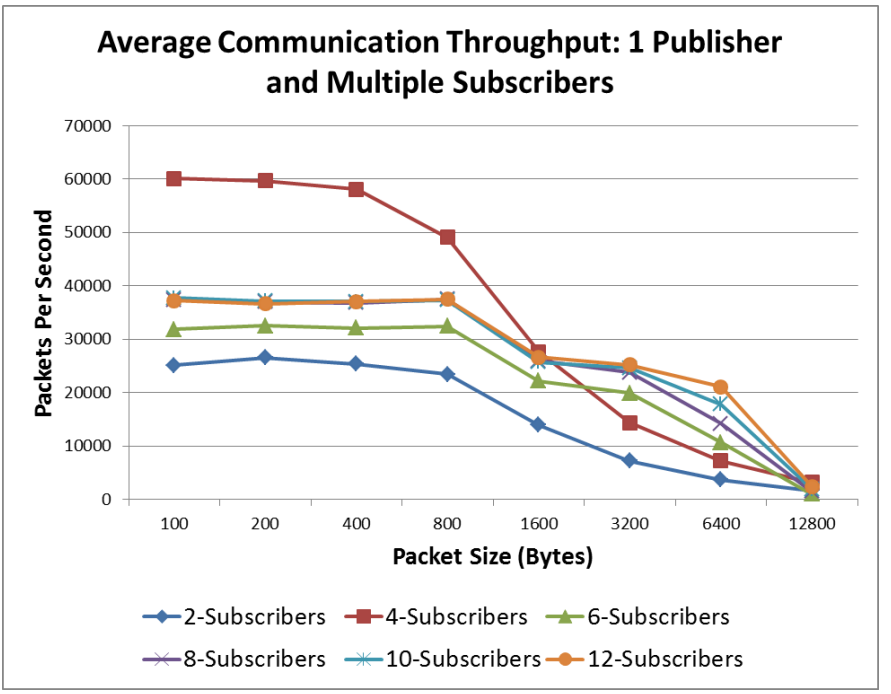

Figure 20 – Average Communication Throughput Performance (Packets/Second)

Figure 21 shows the measured average communication throughput in Megabits per second. The communication throughput increases proportionally to the size of the packet until it reaches the maximum bandwidth available in the Ethernet switch. Each communication link is limited by 100 Mbps. Therefore, for two subscribers, the communication throughput ramps up until it approaches 200 Mbps with packet size of 6400 bytes and starts to drop down after that. Similarly, for twelve subscribers, the communication throughput ramps up until it approaches 1200 Mbps with packet size of 6400 bytes and starts to drop down after that.

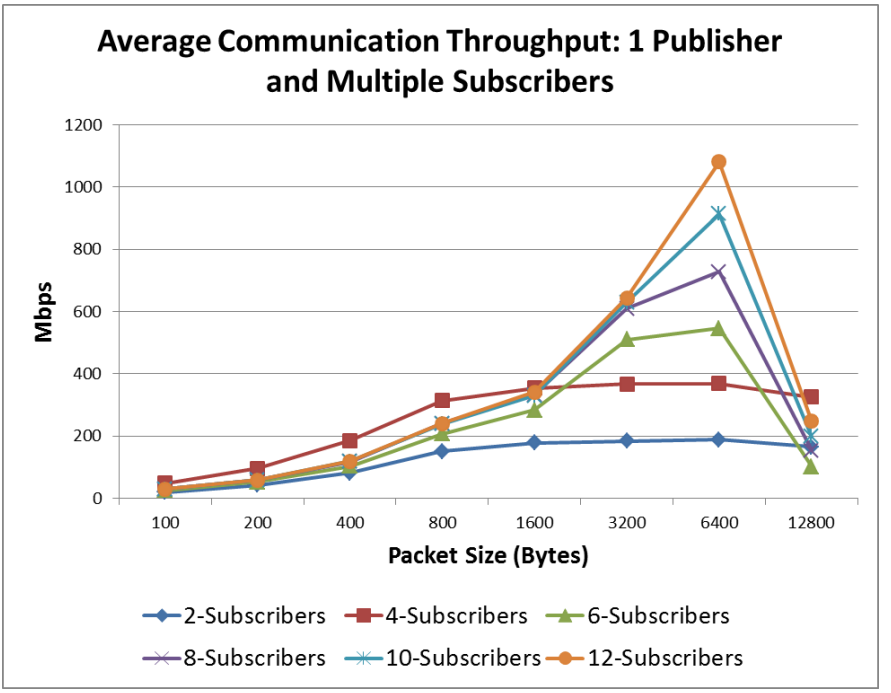

Figure 21 – Average Communication Throughput Performance (Mbps)

7. Conclusion

In this paper we address the limitation in client-server communication model deployed in refining and petrochemical industries for integrating highly interacting process control applications across multiple network layers utilized in distributed control systems. To achieve this objective, this paper presents a new design of collaborative automation platforms for an optimum process control environment to ensure optimum operation of processing equipment for achieving maximum yield of all manufacturing facilities.

This change is accomplished by employing 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 I/O systems and the control applications including sequential and regulatory control, advanced regulatory control, multivariable control, unit-based process control, and plant-wide advanced process control.

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 controller with 400 digital I/O and 50 analog I/O, to a controller size of 12,800 bytes equivalent to a controller 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, making it suitable for scalable collaborative automation platforms.

The following are additional advantages of the collaborative automation platform:

- It is a cost effective evergreen solution because it is based on interoperable and standard COTS software and hardware components resulting in optimum capital investment for the total cost of ownership throughout its life span.
- It is based on a distributed architecture where I/O modules, CPU and control application are not interwoven. Changing I/O voltage signals does not have any impact on the control application.
- It reduces initial capital investment for grass root projects since it does not require any PIBs as well as the system and marshaling cabinets and the associated cable wiring down to the junction boxes.

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. W. Bolton, *Programmable Logic Controllers* (Newnes, 2005)
26. W. Heinzelman, A. Murphy, H. Carvalho and M. Perillo, *Middleware to Support Sensor Network Applications*, IEEE Network Magazine, **18** (2004)
27. 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 currently working on the PhD Dissertation in Computer Science and Engineering at KFUPM. 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.