Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 863123, 11 pages
http://dx.doi.org/10.1155/2015/863123

Research Article

Performance Evaluation of DDS-Based Middleware over Wireless Channel for Reconfigurable Manufacturing Systems

Basem Almadani,[1] Muhammad Naseer Bajwa,[1] Shuang-Hua Yang,[2] and Abdul-Wahid A. Saif[3]

[1]*Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia*
[2]*Department of Computer Science, Loughborough University, Leicestershire LE11 3TU, UK*
[3]*Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia*

Correspondence should be addressed to Muhammad Naseer Bajwa; g201203680@kfupm.edu.sa

Received 9 February 2015; Revised 3 July 2015; Accepted 7 July 2015

Academic Editor: Rob Brennan

Reconfigurable manufacturing systems (RMS) are rapidly becoming choice of production and manufacturing industry due to their quick adaptability to the ever-changing market demands while maintaining the quality and cost of the products. Such systems are usually decentralized in their monitoring and control and consist of heterogeneous components. Therefore, need arises for an interface that can mask the heterogeneity and provide smooth communication among these dissimilar components. Data Distribution Service (DDS) is a data-centric middleware standard based on Real-Time Publish/Subscribe (RTPS) protocol that fulfills the job of such interface in distributed systems. In this work, we present the idea of using DDS-based middleware over commonly used wireless channels like Bluetooth and Industrial WiFi to facilitate data communication in distributed control systems. A simulation model is developed to quantify various performance measures like latency, jitter, and throughput and to examine the suitability of aforementioned wireless channels in distributed monitoring and control environments. The model explores various communication scenarios based upon a practical case study. Obtained results serve as an empirical proof of concept that DDS can ensure reliable and timely data communication in firm real-time distributed control systems using common wireless channels and offer extensive control over various aspects of data transmission through its rich set of QoS policies.

1. Introduction

Use of distributed control systems (DCS) has become quite ubiquitous in a variety of industries like oil refining, petrochemical, food processing, cement production, pharmaceutical, and so forth. Modern controllers have become powerful enough to collect data, make decisions, and issue commands on their own instead of routing the data to a master control unit as in centralized control systems. The most prominent advantages of DCS are their flexibility, agility, adaptability, and no single-point-of-failure. However, this distributed control paradigm raises its own challenges that need to be addressed before optimum benefits could be harvested.

One of these challenges is that DCS not only require I/O communication for every controller but also need horizontal (with other controllers on the same hierarchical level) and vertical (with other devices on different hierarchical levels) communication [1]. Another challenge is the heterogeneity [2] in various components used to constitute the DCS. These components, normally provided by different vendors, vary in their capabilities, data formats, mapping schemes, and I/O interfaces and, therefore, present a rather complex heterogeneous system to deal with.

To facilitate the industrial control communication in DCS and mask the heterogeneity amongst the subsystems, various middleware technologies have been proposed over the last couple of decades. Among these are Web Services, CORBA (Common Object Request Broker Architecture), Java RMI and OPC (OLE for Process Control), and so forth. These technologies can simplify the design significantly and integrate control devices despite their heterogeneity. However, these solutions lose their value in real-time environments

because of their inability to adapt to certain characteristics of real-time process data [1] like periodic messages with data sampled values. Furthermore, they are nondeterministic and do not usually respect strict timelines.

Object Management Group (OMG) developed Data Distribution Service (DDS) as an open and platform-independent middleware standard that uses Real-Time Publish/Subscribe (RTPS) protocol. DDS-based middleware deploys many-to-many communication model and offers extensive control over a large spectrum of Quality of Services (QoS) policies [3]. Although DDS is a relatively new middleware specification, it is gaining substantial attention from researchers as a suitable solution in mission critical industrial automation applications [4–6].

Wireless channel seems to be the natural choice as a communication medium in such reconfigurable environments because wired media will severely limit physical reconfiguration of the system components. Assuming that reconfigurable manufacturing systems normally span a smaller area (the assumptions is generally true for small-scale enterprise), a short-range limited-bandwidth wireless channel like Bluetooth, Zigbee, or WiFi could be a suitable option. For this work Bluetooth and Industrial WiFi have been selected as the communication channels and we intend to measure the performance of DDS over these channels for the control data. The choice of these two channels is inspired by the fact that they are more ubiquitous and mature technologies than contenders. They have relatively higher data rates than other commonly used RF based technologies like Zigbee and are able to support UDP/IP and TCP/IP on transport layer. This later ability is particularly important when integrating them with RTPS protocol of DDS-based middleware.

The rest of the paper is organized as follows. Section 2 briefly explains the fundamentals of DDS and Section 3 outlines some of the recent related works to show the popularity of DDS in distributed industrial control applications and the relevance of this work to published literature. In Section 4, application of Industrial WiFi and Bluetooth in distributed control will be discussed. A mathematical model of RTPS paradigm presented in Sections 5 and 6 discusses a case study for experimentation along with performance measures to analyze the performance of DDS. In Section 7 we will explain the experimental setup and QoS values used to evaluate the proposed approach. The results analysed in Sections 8 and 9 conclude the discussion with a hint of potential future work.

2. Fundamentals of DDS

Middleware technology has been in use for the last four decades or so [8], at first explicitly as stand-alone components and later implicitly as built-in modules, in various roles and shapes. Most of the early models like CORBA [9], OPC [10], Java RMI [11], and Web Services [12] use either client-service communication model or message passing communication paradigm and hence prove somewhat ineffective in real-time environment. They are also platform-dependent. To make middleware more acceptable in real-time and heterogeneous applications, OMG released Data Distribution Service [13]

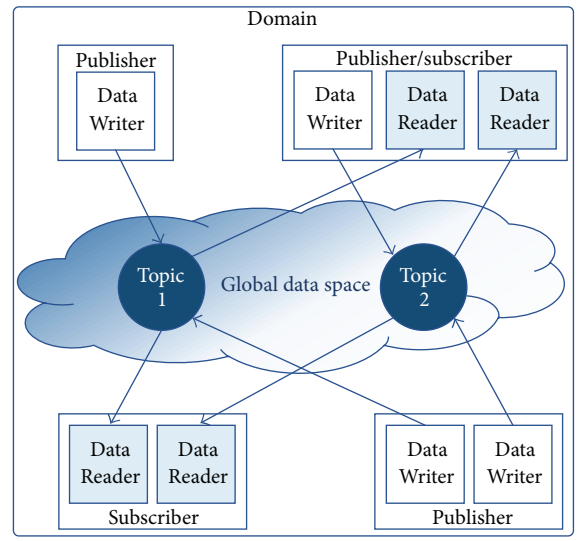

Figure 1: DDS core entities and their relationship in DCPS.

as a platform-independent real-time publish/subscribe middleware standard capable of implementing a broad set of mechanisms to define and manage QoS requirements. Based on these exhaustively elaborative specifications, many implementations are developed that enable various programming language to be combined with commonly used general purpose operating systems. Some open source implementations of DDS include Open DDS and OpenSplice whereas Connext is a proprietary implementation by Real Time Innovation Inc.

DDS employs two levels interfaces which are as follows:

(i) DCPS: Data-Centric Publish Subscribe is the lower level of the two interfaces and is responsible for efficient delivery of proper data to the concerned receiver.

(ii) DLRL: Data Local Reconstruction Layer is an optional higher-lever interface, which allows integration of the middleware and the application layer.

DCPS ensures predictability and high performance of the middleware and the efficient use of the system resources. DLRL automatically reconstructs the data based upon the updated values and gives the application an impression as if the data were local. In this way, the middleware becomes able to transmit the data to all participating subscribers as well as update a local copy of the information. DDS offers a great deal of flexibility and scalability to the distributed systems by effectively decoupling the publishers and subscribers from each other. Conceptual Outline and the basic constructs used by DCPS in the information flow are shown in Figure 1. Brief description of each of them is given in the appendix.

3. Related Work

This section highlights some of the works involving middleware for real-time distributed environment and discusses the suitability of DDS standard in industrial automation scenarios.

The importance of the middleware technology in distributed systems has been greatly emphasized in [8]. The authors have presented brief and comprehensive history of the middleware and have narrated the evolution of the technology right since its inception in the early 1970s. It was regarded as an inevitable component of any large-scale distributed system. In their words, "trying to build a distributed application without middleware is like trying to write a simple application on a personal computer without the operating system." The authors foresee that the technology will become even more indispensable for distributed heterogeneous system as the systems will tend to grow more and more complex in the days to come.

A comprehensive study of trends and developments in wireless manufacturing industry is conducted by Huang et al. in [14]. It has been established that wireless technology in reconfigurable manufacturing can facilitate collecting real-time plant data, improving inventory control and planning, and scheduling and executing the production adaptively as a response to changing customer demands. Their survey has found the application of wireless manufacturing in a broad array of roles like part fabrication, product assembly, Just-in-Time (JIT) manufacturing, and reconfigurable production lines among others.

Realizing that the next generation real-time distributed systems will be highly complex high-performance environments consisting of large number of nodes with heterogeneous characteristics, the authors in [15] proposed an approach to reduce the complexity by using iLand which is an enhanced version of middleware for real-time configuration of service-oriented distributed systems [16]. It has been recognized that future real-time reconfigurable distributed systems are expected to offer data-intensive capabilities by means of assimilating the processing power of large number of nodes. The systems are projected to have increased dynamic behaviour as a result of recurrent reconfigurations, for instance. The approach presented in this work involves modelling the reconfiguration of the system as graphs containing all tentative solutions and finding a new valid solution from the complete graph every time the system undergoes reconfiguration. The results show that solution provides phenomenal decrease in the computation time of the reconfiguration process.

Zhang et al. [17] discussed the possibility of an RFID enabled real-time manufacturing with reconfigurable properties. They proposed a framework of reconfigurable information infrastructure for manufacturing companies. The goal is to enable the manufacturer to implement real-time and smooth dual-way connectivity between RFID enabled devices and the application system at enterprise and shop floor level. They modelled a production process as a workflow network with nodes representing the work and edges corresponding to the data and flow of control. The authors claim that their framework can allow wireless manufacturing data collection and reconfiguration in real time. However, the limited range of RFID technology may pose severe limitation on the span and topology of shop floor.

Although DDS is a very powerful and flexible technology, it may prove to be rather complex to fully comprehend particularly for novice users. This issue was spotted and dealt with in [4]. The authors floated the idea of a software component encapsulating the functionality of commonly used industrial automation controllers like PLCs, IPCs, and Robots which can then be used to create any automation application. The role of DDS in this case can simply be as a communication backbone. The paper shows how to map different traffic patterns using DDS entities taking full advantage of DDS QoS policies.

In [18] Al-Madani et al. conducted performance enhancement of limited-bandwidth wireless industrial control system. They carried out experiments to study various performance measures of control data communication using DDS over Bluetooth and LAN. However, there was no experimental or software model provided in this work. So it cannot be determined how the obtained results will be affected by spatial or topological variations in the physical system.

Large-scale mobile networks find their applications in a variety of areas like emergency response, logistics, transportation management, environmental monitoring, and so forth. These systems normally require real-time tracking of all the nodes and some means of interaction among them. The use of DDS-based middleware in such large-scale mobile networks has been studied by the authors in [19] in which they have customized middleware, termed as Scalable Data Distribution Layer (SDDL), derived from DDS specifications for online tracking and monitoring of large-scale mobile vehicle fleet spread over vast geographical area. SDDL uses two communication protocols: RTPS protocol for communication with SDDL core network over wired media and Mobile Reliable UDP for wireless communication between core network and mobile nodes. The results confirm that the proposed middleware supports mobile nodes handover and multicast and broadcast communication models in real time with acceptable round trip time delays.

Finally, the suitability of short-range limited-bandwidth communication channels, like Zigbee and Bluetooth, in industrial applications has been studied in [20]. The author discussed various merits and demerits of both technologies on different grounds. It is established that because of relatively greater data rate and faster active slave channel access, Bluetooth is better than Zigbee for machine-to-machine communication and for ad hoc connectivity between the fixed equipment and mobile devices.

4. Distributed Control over Wireless Channel

Ethernet is widely used in distributed industrial control as the communication channel due to its high-bandwidth and minimal packet loss. However, wired media are not an option when considering RMS. To avoid unnecessary wiring and make the system more tidy, Industrial WiFi can be used as a wireless substitute for Ethernet; however, it is prone to packet drop with the increase in traffic because it relies upon an access point to transmit the data between communicating nodes. Bluetooth, on the other hand, uses mesh topology that eliminates need of any such device and

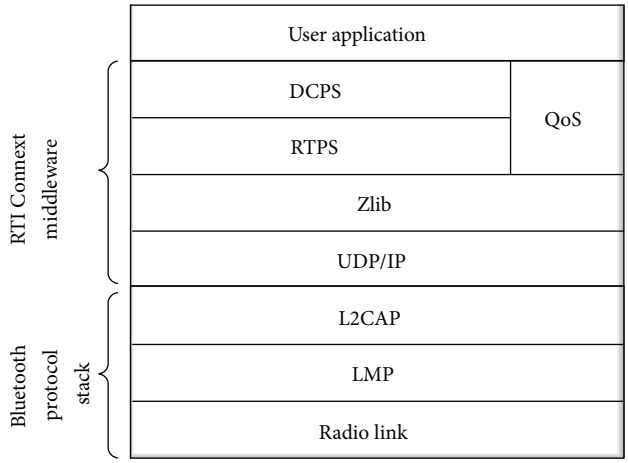

Figure 2: Layered architecture of DDS over Bluetooth.

offers better reliability in terms of data delivery. Though it has comparatively narrower bandwidth, it can still work pretty fine in the given area of application knowing that data-rate requirements in industrial sensing and control applications are often low to intermediate [20].

The RTPS protocol is specifically designed to run in multicast mode over connectionless best-effort transports like UDP/IP; it can also run over connection-oriented reliable transports like TCP/IP. Therefore, integrating DDS with WiFi does not require any tethering or protocol interface between RTPS and WiFi. RTPS runs like "plug and play" over WiFi. With Bluetooth, however, it requires IP to run over Bluetooth so that RTPS can be integrated with UDP/IP or TCP/IP transport protocol [21]. Figure 2 illustrates the layered architecture of DDS over Bluetooth. In the transport layer of DDS, UDP/IP provides the middleware with fine-grained control over data transmission and allows it to decide whether the transmission should be reliable or best effort, depending upon the application requirement and underlying network type.

Above UDP/IP layer is Zlib which is a software library that performs data compression. It is an example where RTI DDS developers can implement their very own data compression algorithms to suit their application specific needs. RTPS wire protocol uses a packet header size of 56 bytes which includes timestamps and submessage headers [13]. This metadata, or transmission overhead, can further be reduced by the governing application, though this area has not been greatly explored yet, especially for low bandwidth channels.

The purpose of this work is to empirically show that DDS-based middleware can meet real-time reliable and efficient data transmission requirements in RMS environment over abovementioned wireless channels and sustain practical QoS support in majority of applications. In the remaining part of this section, we present a test case setting corrosponding to distributed real-time control where judicious selection of QoS policies can lead to smooth, reliable, and on-time data transmission amongst various components of the system.

5. Mathematical Model of RTPS Paradigm

In this subsection, we will briefly discuss the mathematical model of simple publish/subscribe (PS) communication, which is at the heart of DDS standard. Zhai et al. [22] have elaborated the interaction of various entities in PS model with one another using Set Theory. Let we have three actors participating in PS system: publisher, subscriber, and Information Repository. Publishers and subscribers are already explained earlier. Information Repository is responsible for defining acknowledgements. There are three types of objects: notification, subscription, and acknowledgements. Publisher issues notification about its publication and subscriber defines its requirements for subscription. If the notification about a certain publication matches any of the subscription requests, that publication is delivered to the requesting subscriber.

Let B be a 6-tuple defined as

$$B = (P, S, I, N, U, A), \tag{1}$$

where P is a set of publishers, S is a set of subscribers, I is a set of Information Repositories, N is a set of notifications, U is a set of subscriptions, and A is a set of acknowledgements.

B sketches the structure of publish/subscribe system and marks the boundaries of system's state space. The three entities present in the system interact with one another by performing certain action. We define an *event* as an action that changes system's state. Naturally, PS system behaves as discrete event systems. Therefore, we let $E = \{e_1, e_2, e_3, \ldots, e_i, \ldots\}$ be a set of, possibly, infinite events that may occur. Every event e_i occurs at a discrete point in time represented by $t(e_i)$. No two events can occur simultaneously; that is, if $t(e_i) = t(e_j)$, then $e_i = e_j$. Hence, we can only have an ordered sequence of events in which e_i always precedes e_j if and only if $t(e_i) < t(e_j)$ given that $i < j$.

In very primitive model of a PS system the following types of events can occur.

Publish. A notification is published by publisher.

Notify. Subscriber is notified about publication.

Subscribe. Subscriber activates a subscription.

Unsubscribe. Subscriber revokes an existing subscription.

Acknowledge. Information Repository issues acknowledgement.

Having laid down the bases for nomenclature, now we can mathematically define a publish/subscribe as $PS = (B, E)$, where B describes the structure of the system and E determines its behaviour. PS system behaviour can now be modeled as ordered sequence of events which results in change of system states. System state changes when, as an effect of certain event, any of the individual publisher, subscriber, and Information Repository changes its state. For example, when a publisher, P_i, issues a new notification, it changes set of notifications associated with this publisher, $N(P_i)$, triggering a transition in P_i's state. This change in $N(P_i)$

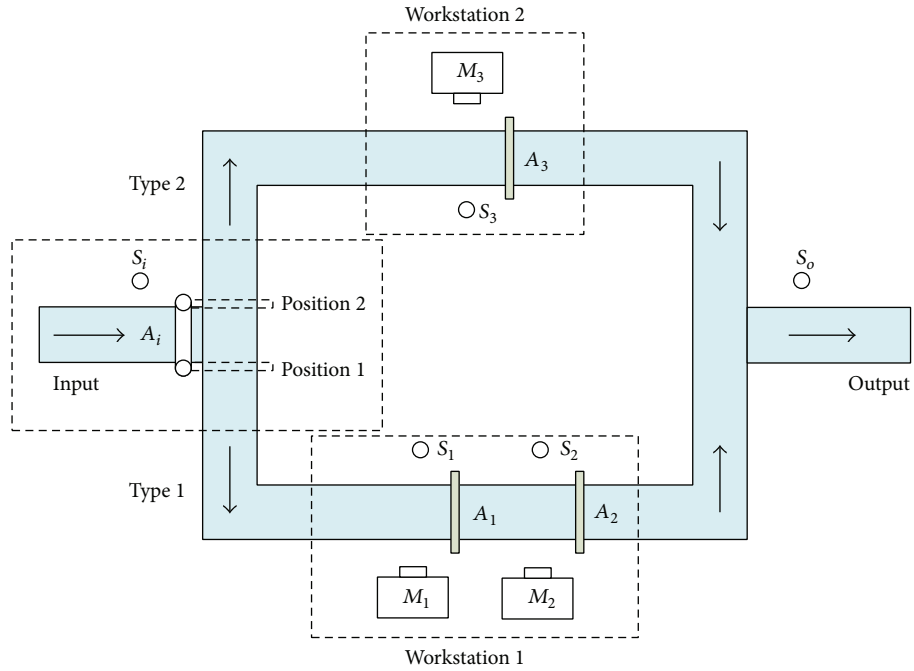

Figure 3: An automated reconfigurable manufacturing system (reproduced with permission from [7]).

ultimately causes change in $N(P)$ which is super set of $N(P_i)$ and represents set of all notifications issued by all publishers in the system. In the same way the state of a subscriber S_j changes as a result of changes in $N(S_j)$ and $U(S_j)$; and any change in $R(P_i)$ causes change of state of Information Repository I_s.

6. An Automated RMS

To evaluate the performance of DDS-based middleware in RMS, consider the automated manufacturing system shown in Figure 3. The system is adapted from [7] and consists of three machines represented by M_1, M_2, and M_3; it has five sensors (S_1, S_2, S_3, S_i, and S_o) and four actuators (A_1, A_2, A_3, and A_i) for blocking. The manufacturing system is constructed using PLCs. Two types of pallets, Type 1 and Type 2, are input to the system randomly. The route which these pallets may take is decided by a three-position stop, A_i, whose selection depends upon the routing information provided by S_1 and S_3 sensors.

All three machines publish a set of values pertaining to their operation. These values include state of the machine and various physical quantities like motor speed, pressure, temperature, and so forth. Machines can be instructed to physically reposition themselves using configuration commands sent by the control panel (not shown in the diagram). Besides issuing configuration commands, control panel also monitors machines' data by subscribing to their respective Topics.

Sensors 1, 2, and 3 look for the availability of their respective machines. If a machine is currently operating on some pallet, then these sensors publish FALSE Boolean value to indicate that the machine is currently busy and no more input may be dispatched on this route. Otherwise, they transmit TRUE as soon as the machine is ready for the next input. Input sensor (S_i) keeps an eye on input availability to the system. Output sensor (S_o) monitors whether a finished product by either workstation has left the system. It helps avoid product collision over the conveyor belt.

Actuators 1, 2, and 3 are stop points and remain close as long as corresponding machines are working. They are open to let the finished product pass and let new input get into the workstation. Dispatcher A_i is responsible for forwarding the input (if available) to either of the two routes or hold until one of them is available.

The motors, actuators, and sensors can physically move across the workbench (shown in grey strip) enabling the whole system to transform into a single-, double-, or triple-workstation system, thus facilitating configurability of the system in both physical and functional sense. As it is assumed that the whole setup covers a relatively smaller space (up to few tens of meters), therefore use of limited-bandwidth wireless channel seems proper in this scenario.

6.1. Performance Measures. Before moving on to the experimentation, it is better to first discuss the performance measures that will determine whether Bluetooth and WiFi can fulfil real-time data communication requirements of the system.

(i) Latency: latency is the time a data packet takes to reach the receiver side. It includes propagation delay plus queuing delay at the receiver side. It can be

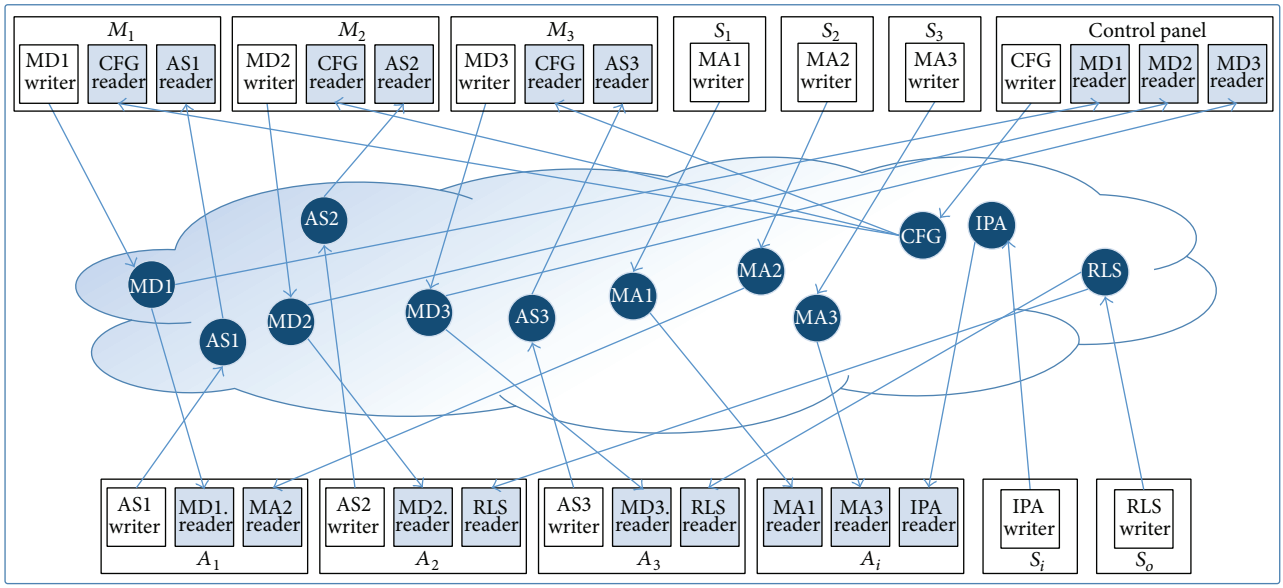

FIGURE 4: DCPS model of system shown in Figure 3. MDx = Machine Data x, CFG = configuration command, MAx = Machine Availability x, ASx = Actuator Status x, RLS = Relay Status, IPA = input availability, MDx. = Machine Data x. Status (status member of data structure MDx).

calculated by dividing round trip time (RTT) by 2 as given in

$$\text{Latency} = \frac{RTT}{2}. \tag{2}$$

In firm real-time systems, deadlines are relatively relaxed as compared to hard real-time systems. Although a packet arriving after the deadline may not be of any value, occasional longer delays or even packet loss does not cause the system to fail [23, 24].

In this experimentation, every data packet is sent with a time stamp according to publisher's clock. The publisher upon receiving the acknowledgement for the packet calculates the time difference between sending the packet and receiving the acknowledgement and marks the duration as RTT. One way latency is therefore taken as half of RTT.

(ii) Jitter: jitter is the variation in latency. Smaller values of jitter mean that the data will, most of the time, experience almost the same amount of delay during its voyage from sender to receiver. Mathematically, it can be represented as given in

$$\text{Jitter} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}. \tag{3}$$

Here, N is the total number of delay samples and $\overline{x}$ is the mean value of N delay samples. Jitter is significant in determining the precision of the system. If a channel shows sufficiently small average latency during a transmission session but exhibits large jitter values, it signifies that some packets take unusually long time to reach the subscriber and any given update cannot be reliably sent within average latency duration.

(iii) Throughput: throughput denotes average rate of successful data transmission over a channel. This data rate does not take only payload into consideration but also includes any protocol overhead. We used (4) to calculate the throughput:

$$\text{Throughput} = \frac{\text{Packet Size} \times \text{number of Packets}}{\text{Total Time}}. \tag{4}$$

Throughput is important in observing channel utilization. High throughput implies that bandwidth resources are property utilized. However, increasing throughput beyond a certain point may cause congestion and subsequently result in packet loss. This in turn may cause longer delays. Therefore, monitoring throughput is pivotal in real-time mission critical applications. Bluetooth offers several frame formats with varying header and payload sizes. The most common format has 126 bits of metadata and up to 2744 bits of payload. However for a single time slot of 625 μs (there are 1600 frequency hops per second in Bluetooth) the frame carries only 240 bits as payload, while frame metadata remains the same [25, 26].

7. Experimental Setup

Simulation model corresponding to the scenario depicted in Figure 3 is shown in Figure 4. As can be seen from the figure, the model incorporates one-to-many and many-to-many communication requirements. Each rectangle and each square represent a domain participant (we are assuming that

Table 1: QoS used in experimentation.

QoS Policy	Value	
	Publisher	Subscriber
DURABILITY	TRANSIENT	TRANSIENT
LATENCY_BUDGET	40 ms	40 ms
LIVELINESS	AUTOMATIC	AUTOMATIC
max_lease_duration	120 sec	120 sec
RELIABILITY	BEST_EFFORT	BEST_EFFORT
HISTORY	KEEP_ALL	KEEP_LAST
RESOURCE_LIMITS *max_samples_per_instance*	LENGTH_UNLIMITED	1

all the participants are in a single domain). These DDS entities correspond to various types of hardware devices like sensors, actuators, and motors as shown in Figure 3. The QoS used in experimentation are given in Table 1.

The *rtiddsgen* utility provided by RTI Connext 5.0.0 is used to generate C++ code. Visual Studio 2010 is used to build the code and Wireshark 1.2.3 is employed for traffic monitoring over wireless channels.

7.1. QoS Policies Used in Experimentation. Below, we present a short description of the QoS used for the experimentation and their interdependence on one another.

(i) *DURABILITY*: durability QoS decides if data should outlive their writing time, that is to say, whether or not data samples should be archived in middleware service after they are written. A VOLATILE type does not care to save any sent data samples on behalf of Data Writers; however, TRANSIENT type maintains record of sent updates in memory and the data is not tied to the lifecycle of Data Writer. It means that data will still be available even if the corresponding Data Writer goes offline. These achieved samples may be delivered to late-joining subscribers who want to know what they missed.

(ii) *LATENCY_BUDGET*: this QoS policy describes maximum acceptable delay between sending and receiving of the data. This is not something carved in the stone, but rather just a guideline to the service. If an update fails to meet this acceptable delay, the service will not raise any flags or discard the packet.

(iii) *LIVELINESS*: it indicates the mechanism by which the middleware knows if any participating entity is active or has gone offline. Every Data Writer periodically signals its liveliness to all the Data Readers. The signalling period must not exceed *liveliness_lease_duration*; otherwise Data Reader assumes that the Data Writer is no more alive.

(iv) *RELIABILITY*: the RELIABILITY QoS policy specifies the level of reliability that a subscriber can offer or a publisher can request. It has two values, RELIABLE and BEST_EFFORT. In RELIABLE mode, the Data Reader must acknowledge the receiving of each and every packet that arrives. Data Writer does not discard any data value that has been transmitted but not yet acknowledged. This approach has slightly negative effect on the latency because the receiving entity must check the integrity and order of the received packet before acknowledging. It also consumes some of the channel bandwidth for acknowledgements. On the other hand, if a packet drop, every once in a while, does not greatly affect the system and the application cares little about the order of the data received, it is better to use BEST_EFFORT mode which does not require sending or receiving acknowledgements.

(v) *HISTORY*: this QoS defines the behaviour of middleware service in case the value of the data changes before it is successfully transferred to the receiver. On sender side, it controls the number of samples that will be kept with Data Writer on behalf of Data Reader. On receiver side, it indicates the number of samples maintained by Data Reader until the subscriber application reads the data.

(vi) *RESOURCE_LIMIT*: it indicates how much resources the middleware may consume to comply with the QoS requirements. Configuration of this QoS must be in accordance with other QoS settings. For example, if RELIABILITY is set to RELIABLE, then Data Writers need some memory space to store the data samples that have been sent but not yet acknowledged. If we set *max_samples_per_instance* to, say, 1, then Data Writer will not be able to cache enough unacknowledged packet to implement RELIABLE communication. Therefore, to implement RELIABILITY QoS successfully, enough resources must be allocated using RESOURCE_LIMIT.

8. Results and Analysis

The simulation model mimics the behaviour of communicating devices; it generates random data values periodically and publishes them. It also plays role of subscribing components. This simulation model runs on different computer machines connected via Bluetooth or WiFi. The generated values are transmitted over physical channel and actual measurements are taken and recorded for analysis.

Table 2: Latency and jitter for one-to-many scenario.

Setup	Min. (m sec)		Max. (m sec)		Average (m sec)		Jitter (m sec)	
	BT	WiFi	BT	WiFi	BT	WiFi	BT	WiFi
1-1	7.94	1.28	50.30	63.51	9.11	2.41	4.60	2.35
1-2	11.00	1.38	71.08	57.29	12.62	2.47	5.46	1.96
1-4	17.31	1.77	64.08	112.44	19.42	4.00	6.71	4.68
1-6	23.54	2.06	72.50	454.14	26.16	4.23	7.67	5.16
1-8	29.79	2.22	99.56	707.40	33.45	5.95	8.84	7.63

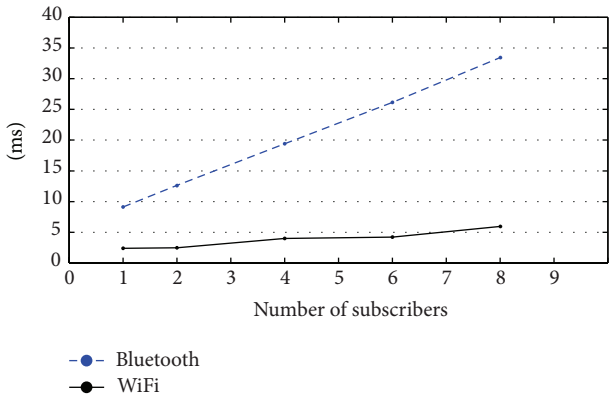

Figure 5: Average latency against various number of subscribers.

8.1. Experiments for Latency and Jitter. For the latency and jitter test, we used payload of 1024 bytes (which is the maximum payload size in the given scenario). First one publisher and multiple subscriber scenario is examined and latency and jitter are calculated. In each run 10,000 to 50,000 packets are sent; the tests are repeated up to 10 times and average is taken to make the results more precise. Table 2 shows the obtained results.

Based upon the collected data, jitter is calculated using (3). We can see that both latency and jitter increase linearly as the number of subscribers grow. Due to higher bandwidth, WiFi latency is significantly lower than Bluetooth latency, despite the fact that WiFi packets from one node must go to an access point before being routed to the destination node. However, the values of average latency for Bluetooth, even for the worst case, are well within acceptable range for firm real-time requirements (around 40 msec).

Figures 5 and 6 show the graphs of latency and jitter corresponding to Table 2, respectively. We also observe that while WiFi jitter is far smaller than Bluetooth jitter for fewer subscribers, the gap between the two tends to shrink as more and more subscribers (and consequently network traffic) join in and packet drop increases. This is because of the fact that WiFi performance at a certain instance depends greatly on the amount of traffic at that given point in time.

Latency and jitter are calculated for many-to-many communication scenario as well. Tests are run to examine the effect of multiple publishers and multiple subscribers on the latency and jitter. As expected, both performance measures have higher values when multiple participants try to transmit the data over a single channel. These results are tabulated in Table 3. Figure 7 shows the results in graphical format. Here again we can see that DDS ensures small enough latency and jitter over both communication channels to accomodate firm real-time requirements of most RMS. However, WiFi precedes Bluetooth in terms of better latency and jitter in every scenario.

In these experiments, the maximum number of subscribers is limited to 8. This is for two reasons: firstly, our case study does not require more than 8 subscribers for any control data and secondly the trend in the performance measures can be easily deduced with these many participants. It is anticipated that with the increase in the number of

Table 3: Latency and jitter for many-to-many model.

Setup	Min. (m sec)		Max. (m sec)		Average (m sec)		Jitter (m sec)	
	BT	WiFi	BT	WiFi	BT	WiFi	BT	WiFi
2-2	8.62	1.42	167.90	131.93	18.34	2.69	8.94	2.95
2-4	11.10	1.64	210.58	237.24	19.31	3.43	7.54	3.36
4-4	15.94	1.55	251.02	106.48	23.71	3.40	12.31	3.39
4-8	22.15	2.43	259.27	291.53	37.05	6.32	9.88	4.28

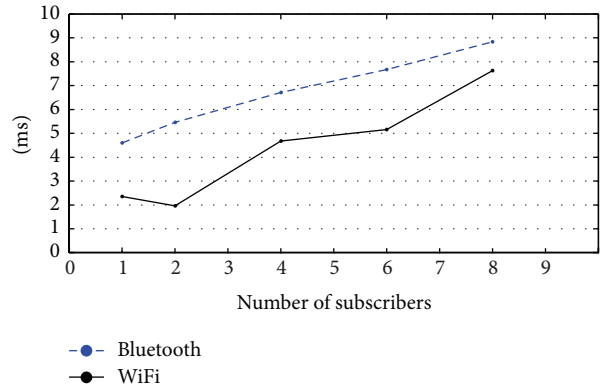

Figure 6: Jitter against various number of subscribers.

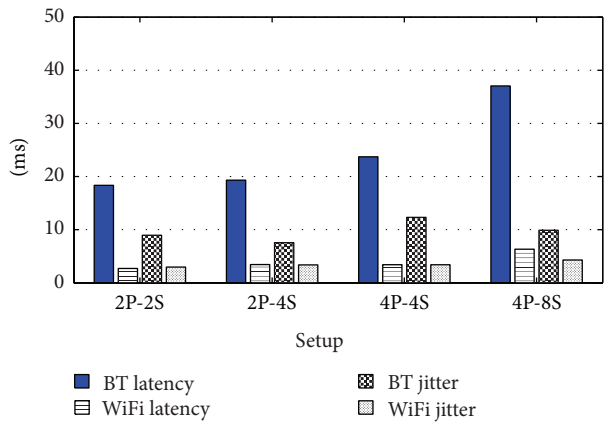

Figure 7: Average latency and jitter for various many-to-many scenarios.

Table 4: Throughput for one-to-many model.

Setup	Total packets (×1000)		Total time (sec)		Throughput (Mbps)	
	BT	WiFi	BT	WiFi	BT	WiFi
1-1	3137	4585	1107	5515	23.22	6.81
1-2	3091	2794	1125	3288	22.50	6.96
1-4	3156	3330	1110	3805	23.28	7.17
1-6	3560	3780	1235	3985	23.61	7.77
1-8	3786	5040	1311	5505	23.66	7.50

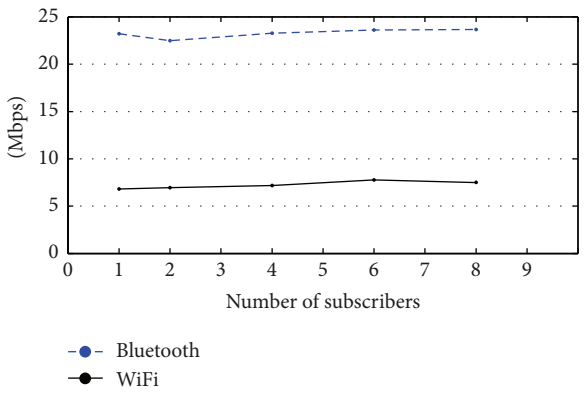

Figure 8: Throughput graph for single publisher and multiple subscribers.

Table 5: Throughput for many-to-many model.

Setup	Total packets (×1000)		Total time (sec)		Throughput (Mbps)	
	BT	WiFi	BT	WiFi	BT	WiFi
2-2	3368	3560	1238	3706	22.28	7.87
2-4	4082	4032	1399	4360	23.90	7.67
4-4	4178	4032	1445	2767	23.69	12.37
4-8	4208	4032	1445	2730	23.86	12.59

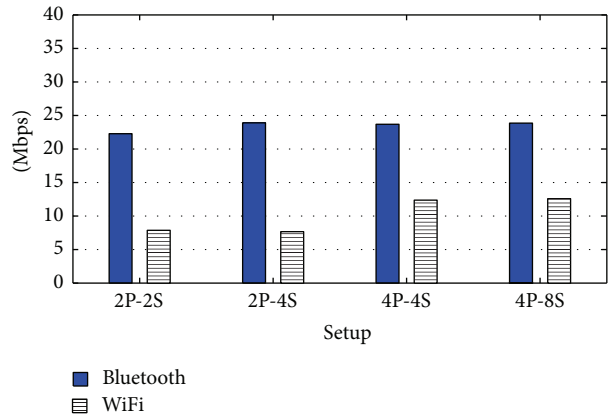

Figure 9: Throughput graph for multiple publishers and multiple subscribers.

subscribers latency, jitter, and throughput will follow the same course as obtained in these experiments.

8.2. Experiments for Throughput. Throughput depends upon the size of the data packets sent and the frequency of the transmission. Most of the data size used in the case study is less than 150 bytes. Only configuration commands may extend up to 1 KB. We used this maximum packet size to calculate the throughput of the Bluetooth channel. This time, 100,000 to 160,000 samples of data packets are sent from the publisher side and received on the subscriber side in each iteration. Total time for this communication is noted and throughput is calculated using (4). The experiments are conducted at least 10 times to get more precise results.

We notice from Table 4 and Figure 8 that the throughput for the given packet size is not quite affected by the number of participants currently active. There is very small and random difference in the throughput against various number of subscribers. Bluetooth provides extremely high throughput for 24 Mbps channel (for Bluetooth 3.0 + HS). Though the throughput of WiFi is significantly low, it provides better latency values for the low data-rate communication.

Like many-to-many latency and jitter experiments, we also conducted tests to measure average throughput over Bluetooth and Industrial WiFi in many-to-many mode. Various configurations of publishers and subscribers are examined. Table 5 summarizes the results obtained for both channels. We can observe from Figure 9 that throughput is not significantly affected by the number of participants in many-to-many scenarios as well and once again Bluetooth surpasses WiFi in terms of higher average throughput. It should be noted, however, that tests on WiFi require a very controlled environment to avoid any unnecessary traffic and ensure that only DDS applications are using the channel. This is not the case with Bluetooth because of its point-to-point connection.

9. Conclusion and Future Work

This work is a first attempt to investigate the suitability of Bluetooth and Industrial WiFi in RMS applications taking full advantage of DDS-based middleware. It is established that most of the distributed industrial control systems involve exchange of simple control parameters and system states and therefore require relatively low data rates. DDS-based middleware can mediate among heterogeneous devices in a typical small-area RMS by offering a data-centric communication paradigm for abstracting their peculiar data representations. The results show that DDS over these wireless channels fulfil real-time data communication requirements of most of the limited-bandwidth small-area control systems. They offer high throughput for small data packets and low latency which is suitable for firm real-time systems. These performance measures in conjunction with inherent security and reliability of these channels make them a safe and reliable choice for most RMS applications.

Literature survey reveals that not enough attention has been granted to wireless RMS applications and the role of DDS-based middleware. Results obtained in this work are encouraging and call for further focused research to study

the proposed concept in greater depth. For future research work, use of DDS over Zigbee in DCS environment can be an interesting area to investigate. Injecting RTPS data over rather low-bandwidth Zigbee network can be challenging. Research in this area can present proof of concept of Zigbee's suitability in mission critical real-time applications using DDS-based middleware. It is, however, expected that latency and jitter may increase significantly given narrower bandwidth of the channel. The work in this area is currently underway.

Appendix

Figure 1 shows the fundamental construct of DDS. Below, we present brief account of each of them for readers not familiar with DDS architecture.

(i) Topic: a Topic is an information unit that is exchanged across the distributed system. Every Topic is identified by a unique name within a domain and has an exclusive data type.

(ii) Domain: a domain is a communication context. It is a distributed concept that links all those applications that can communicate with each other. In other words, all the participants belonging to a certain domain can interact only to each other and not to the participants in another domain.

(iii) Domain Participants: participants are the entities involved in data communication in any application. They represent the local membership of the application in a certain domain.

(iv) Publisher: a publisher is an object responsible for data distribution. It can publish various types of data using Data Writers. Domain participants create publishers to manage a group of Data Writers.

(v) Subscriber: a subscriber is an object responsible for receiving the published data and making it available for the application that the data are supposed to reach. It can receive different types of data using Data Readers. Domain participants create subscribers to manage a group of Data Readers.

(vi) Data Writers: Data Writer is a typed-based object that the middleware uses to communicate the existence and the value of the data object of a certain type. *Typed-based* means that each Data Writer object is associated to only one data type or Topic. If a publisher wants to publish more types of data it must have dedicated Data Writers for each of the Topics.

(vii) Data Readers: Data Readers are typed-based objects that a subscriber must use to access subscribed data. Each Data Reader can read only a single Topic and different Data Readers need to be attached to a subscriber in order to receive multiple data types. Therefore, a *subscription* may be defined as an correspondence between a Data Reader and a subscriber.

Every time an application wants to publish data it must create a publisher, or use an existing one, and connect it with a Data Writer that matches the characteristics of the desired publication. Similarly, when an application wants to start receiving updates on a particular Topic it must create a Subscriber, or reuse an existing one, and link it to an appropriate Data Reader.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to extend their gratitude to Real Time Innovation Inc. for very generously providing RTI Connext license for this project. Computer Engineering Department of KFUPM also deserves special thanks for arranging hardware equipment and lab setup for the experimentation.

References

[1] I. Calvo, F. Pérez, I. Etxeberria, and G. Morán, "Control communications with DDS using IEC61499 service interface function blocks," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '10)*, pp. 1–4, IEEE, September 2010.

[2] B. Almadani, S. Khan, M. N. Bajwa, T. R. Sheltami, and E. Shakshuki, "AVL and monitoring for massive traffic control system over DDS," *Mobile Information Systems*. In press.

[3] B. Almadani, A. Al-Roubaiey, and R. Ahmed, "Manufacturing systems integration using real time QoS-aware middleware," *Advanced Materials Research*, vol. 711, pp. 629–635, 2013.

[4] I. Calvo, F. Pérez, I. Etxeberria-Agiriano, and O. G. De Albéniz, "Designing high performance factory automation applications on top of DDS: regular paper," *International Journal of Advanced Robotic Systems*, vol. 10, article 205, 2013.

[5] I. Etxeberria-Agiriano, I. Calvo, F. Perez, and G. de Albeniz, "Mapping different communication traffic over DDS in industrial environments," in *Proceedings of the 6th Iberian Conference on Information Systems and Technologies (CISTI '11)*, pp. 1–6, IEEE, Chaves, Portugal, June 2011.

[6] J. L. Poza, J. L. Posadas, and J. E. Simó, "QoS-based middleware architecture for distributed control systems," in *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, vol. 50 of *Advances in Soft Computing*, pp. 587–595, Springer, Berlin, Germany, 2009.

[7] M. Nourelfath, D. Ait-kadi, and W. I. Soro, "Availability modeling and optimization of reconfigurable manufacturing systems," *Journal of Quality in Maintenance Engineering*, vol. 9, no. 3, pp. 284–302, 2003.

[8] J. Al-Jaroodi and N. Mohamed, "Middleware is STILL everywhere!!!," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 16, pp. 1919–1926, 2012.

[9] Object Management Group, "CORBA Specifications, version 3.3," 2012, http://www.omg.org/spec/CORBA/3.3/.

[10] OPC Foundation, 2013, https://opcfoundation.org/.

[11] Oracle Corporation, "Java Platform Standard Edition 8 Documentation," 2014, http://docs.oracle.com/javase/8/docs/index.html.

[12] The World Wide Web Consortium, *Web Service Architecture*, W3C, 2004, http://www.w3.org/TR/ws-arch/.

[13] Object Management Group, "DDS for Real-time Systems, Version 1.2," 2007, http://www.omg.org/spec/DDS/1.2/.

[14] G. Q. Huang, P. K. Wright, and S. T. Newman, "Wireless manufacturing: a literature review, recent developments, and case studies," *International Journal of Computer Integrated Manufacturing*, vol. 22, no. 7, pp. 579–594, 2009.

[15] M. García-Valls, P. Uriol-Resuela, F. Ibáñez-Vázquez, and P. Basanta-Val, "Low complexity reconfiguration for real-time data-intensive service-oriented applications," *Future Generation Computer Systems*, vol. 37, pp. 191–200, 2014.

[16] M. G. Valls, I. R. Lopez, and L. F. Villar, "ILAND: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 228–236, 2013.

[17] Y. Zhang, G. Q. Huang, T. Qu, and O. Ho, "Agent-based workflow management for RFID-enabled real-time reconfigurable manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 2, pp. 101–112, 2010.

[18] B. Al-Madani, A. Al-Roubaiey, and M. F. Al-Hammouri, "Performance enhancement of limited-bandwidth industrial control systems," *Advanced Materials Research*, vol. 739, pp. 608–615, 2013.

[19] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler, "A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes," *Journal of Internet Services and Applications*, vol. 4, no. 1, pp. 1–15, 2013.

[20] N. Baker, "ZigBee and Bluetooth strengths and weaknesses for industrial applications," *Computing & Control Engineering Journal*, vol. 16, no. 2, pp. 20–25, 2005.

[21] U. F. Khan, S. Hameed, and T. Macintyre, "TCP/IP over bluetooth," in *Advances in Computer and Information Sciences and Engineering*, pp. 479–484, Springer, Dordrecht, The Netherlands, 2008.

[22] L. Zhai, L. Sun, and Y. Liu, "Modeling and evaluation of high-performance publish-subscribe system," in *Proceedings of the International Symposium on Computational Intelligence and Design (ISCID '08)*, vol. 1, pp. 457–460, IEEE, 2008.

[23] T. T. H. Le, L. Palopoli, R. Passerone, Y. Ramadian, and A. Cimatti, "Parametric analysis of distributed firm real-time systems: a case study," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '10)*, pp. 1–8, September 2010.

[24] T. T. H. Le, L. Palopoli, R. Passerone, and Y. Ramadian, "Timed-automata based schedulability analysis for distributed firm real-time systems: a case study," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 3, pp. 211–228, 2013.

[25] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, Prentice Hall, 5th edition, 2012.

[26] Bluetooth Special Interest Group, "Bluetooth Specifications Version 4.1," 2013, https://www.bluetooth.org/en-us/specification/adopted-specifications.