4-11-2013

Lightweight Middleware for Software Defined Radio (SDR) Inter-Components Communication

Pasd Putthapipat
Florida International University, pputt001@fiu.edu

DOI: 10.25148/etd.FI13042334

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

LIGHTWEIGHT MIDDLEWARE FOR SDR INTER-COMPONENTS COMMUNICATION

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Pasd Putthapipat

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Pasd Putthapipat, and entitled The Lightweight Middleware for SDR Inter-components Communication, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

_____
Chen Liu

_____
Gang Quan

_____
Deng Pan

_____
Jean H. Andrian, Major Professor

Date of Defense: March 22, 2013

The dissertation of Pasd Putthapipat is approved.

_____
Dean Amir Mirmiran
College of Engineering and Computing

_____
Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

To my Mother and Father, who always make me a better me.

To Patomporn, who always holds my hands.

Without them, this work would not have been possible.

ACKNOWLEDGMENTS

For me, this dissertation value is not just the dissertation to fulfill my graduation, it is also the representation of the holistic combination of knowledge, spirit and, most importantly, love from my beloved ones who helped me to reach this point and complete my dissertation. First and foremost, I would like to offer the credit to my advisor, Associate Professor Jean H. Andrian. Without his guidance, support, and encouragement, I would not be able to complete this task. Next, I would like to present my special thanks to the dissertation committees: Assistant Professor Chen Liu, Assistant Professor Deng Pan, and Associate Professor Gang Quan who have supported my work and provided me essential ideas about them. I would also like to acknowledge Dr. Herman Watson for his kind support and understanding.

Additionally, my gratitude goes to Dr. Shekhar Bhansali, Dr. Kang Yen, Madam Maria Benincasa, and all administrative staffs at the Department of Electrical and Computer Engineering for their kind assistance and support. I would also like to include Dr. Khokiat Kengskool for his support and opportunity.

I would like to express my gratitude to Rev. Bro. Dr. Bancha Saenghiran and the Assumption University for providing the opportunity and financial support for this study as well as Dr. Sudhiporn Patumtaewapibal, Assistant Professor Dr. Kittiphan Techakittiroj, and everyone at the AU School of Engineering for their well-received supports.

Last but not least, my studies and work could not have been possible without the love and support from my parents and my girlfriend that put their trust in me without any

doubt. They always supported me financially, physically, mentally, and spiritually to get through these course of studies.

ABSTRACT OF THE DISSERTATION

LIGHTWEIGHT MIDDLEWARE FOR SDR INTER-COMPONENTS COMMUNICATION

by

Pasd Putthapipat

Florida International University, 2013

Miami, Florida

Associate Professor Jean H. Andrian, Major Professor

The ability to use Software Defined Radio (SDR) in the civilian mobile applications will make it possible for the next generation of mobile devices to handle multi-standard personal wireless devices and ubiquitous wireless devices. The original military standard created many beneficial characteristics for SDR, but resulted in a number of disadvantages as well. Many challenges in commercializing SDR are still the subject of interest in the software radio research community. Four main issues that have been already addressed are performance, size, weight, and power.

This investigation presents an in-depth study of SDR inter-components communications in terms of total link delay related to the number of components and packet sizes in systems based on Software Communication Architecture (SCA). The study is based on the investigation of the controlled environment platform. Results suggest that the total link delay does not linearly increase with the number of components and the packet sizes. The closed form expression of the delay was modeled using a logistic function in terms of the number of components and packet sizes. The model performed well when the number of components was large.

Based upon the mobility applications, energy consumption has become one of the most crucial limitations. SDR will not only provide flexibility of multi-protocol support, but this desirable feature will also bring a choice of mobile protocols. Having such a variety of choices available creates a problem in the selection of the most appropriate protocol to transmit. An investigation in a real-time algorithm to optimize energy efficiency was also performed. Communication energy models were used including switching estimation to develop a waveform selection algorithm. Simulations were performed to validate the concept.

TABLE OF CONTENTS

CHAPTER PAGE

1. INTRODUCTION....1
   1.1. Objective and Contribution....5

2. SOFTWARE DEFINED RADIO....10
   2.1. What is "Software Defined Radio"....10
   2.2. Terminology....10
   2.3. SDR Motivation....12
   2.4. SDR Hardware Architecture....15
   2.5. SDR Software Architecture....16
   2.6. SCA and Middleware problem....24

3. STUDIES OF NUMBER OF COMPONENTS VS LATENCY IN SDR....28
   3.1. Experiment Methodology....28
   3.2. Experimental system setup and Assumption environment....30
   3.3. Results....30
   3.4. Analysis and Modeling....31
   3.5. Conclusion....44

4. STUDIES ON INTER COMPONENT COMMUNICATION LATENCY BASED ON VARIATION OF NUMBER OF COMPONENTS AND PACKET SIZE IN SDR....46
   4.1. Experiment Methodology....46
   4.2. Experimental system setup and Assumption environment....47
   4.3. Results....48
   4.4. Analysis and Modeling....52
   4.5. Conclusion....62

5. ENERGY AWARE WAVEFORM SELECTION ALGORITHM FOR SDR....63
   5.1. Energy Model and Selection Algorithm....63
   5.2. Simulation Setup, Environment, and Performance Index....70
   5.3. Results and Analysis....70
   5.4. Conclusion....81

6. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH....82

REFERENCE....85

VITA....90

LIST OF FIGURES

FIGURE PAGE

1. Block diagrams of sample SDR waveforms and platforms [23] © 2009 by IEEE..... 12
2. SDR Hardware architecture [28] 15
3. SCA software structure [23] 20
4. SCA Management Hierarchy at Instantiation [31] 21
5. Diagram shows where the timestamps are kept..... 30
6. Graph of Average total link delay vs. Number of components 32
7. Graph of Average link delay vs. Number of components 33
8. Generic representation of abstract level waveform communication [31]..... 35
9. Generic representation of actual internal waveform communication..... 36
10. Graph of Actual average total link delay vs. Number of components..... 40
11. Contour plot of least square method..... 41
12. Graph of average total link delay vs. Number of components 42
13. Graph of Absolute error vs. Number of components..... 43
14. Graph of relative error in percentage (%) vs. Number of components..... 44
15. Graph of average total link delay vs. Number of components 45
16. Graph of Absolute error vs. Number of components..... 45
17. Graph of relative error in percentage (%) vs. Number of components..... 46
18. Graph of Average total delay vs. Packet size. The waveform has 2 dummyblock components. 50
19. Graph of Average total delay vs. Packet size. The waveform has 3 dummyblock components. 50
20. Graph of Average total delay vs. Packet size. The waveform has 4 dummyblock components. 51

21. Graph of Average total delay vs. Packet size. The waveform has 5 dummyblock components. 51

22. Graph of Average total delay vs. Packet size. The waveform has 6 dummyblock components. 52

23. Graph of Average total delay vs. Packet size. The waveform has 7 dummyblock components. 52

24. Graph of Average total delay vs. Packet size. The waveform has 8 dummyblock components. 53

25. Graph of Average total delay vs. Packet size. The waveform has 9 dummyblock components. 53

26. Graph of Average total delay vs. Packet size. The waveform has 10 dummyblock components. 54

27. Graph of Average total link delay vs. Number of components for four different packet sizes 54

28. Graph of Average total link delay vs. Number of components for four different packet sizes 56

29. Graph of K vs. Linear value of packet size 58

30. Graph of t_o vs. Linear value of packet size 59

31. Graph of average total link delay vs. Number of components of 4 different packet sizes 60

32. Graph of Absolute error vs. Number of components of four different packet sizes 61

33. Graph of relative error in percentage (%) vs. Number of components of four different packet sizes 62

34. Graph of average total link delay vs. Number of components of 4096 bits packet sizes 63

35. Graph of Absolute error vs. Number of components of 4096 bits packet size 63

36. Graph of relative error in percentage (%) vs. Number of components of 4096 bits packet size 64

37. Energy Aware Module for SCA 70

38. Energy Aware Waveform Selection Scheme 71

39. Energy Consumption vs. Traffic Length 73

40. Energy Consumption vs. Traffic Length zoomed on the first five hundred bytes..... 73

41. Example of Real Life Application Traffic length: Energy Consumption vs. Traffic Length 74

42. Normalized Energy Consumption of short traffic length Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Proposed Scheme 75

43. Normalized Energy Consumption of short traffic length. Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Proposed Selection Scheme .. 76

44. Normalized Energy Consumption with high probability (> 0.75) of long traffic length (> 2000 bytes). Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme 78

45. Normalized Energy Consumption with high probability (> 0.75) of short traffic length (< 200 bytes). Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme 78

46. Smart Energy Aware Waveform Selection Scheme 80

47. Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of long traffic length (> 2000 bytes). 81

48. Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of short traffic length (< 200 bytes)..... 81

49. Normalized Energy Consumption of “normal distribution traffic length (blue)” ,“high probability (> 0.75) of short traffic length (< 200 bytes) (green)”, and “Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of long traffic length (> 2000 bytes) (red)”. Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme with threshold changing 82

ACRONYMS AND ABBREVIATIONS

A/D	Analog to digital
CCA	Clear Channel Assessment
CDMA	Code division multiple access
CORBA	Common Object Request Broker Architecture
CR	Cognitive Radio
CSMA/CD	Carrier sense multiple access with collision detection
D/A	Digital to analog
DCF	Distributed coordination function
DDC	Digital down-convertor
DUC	Digital up-convertor
GIOP	General Inter-ORB Protocol
GPP	General purpose processor
IDL	Interface Definition Language
IIOP	Internet InterORB Protocol
JTRS	Joint Tactical Radio System
MTU	Maximum Transmission unit
OMG	Object Management Group
OTA	Over the air
PIM	Platform independence model
QoS	Quality of Service
SCA	Software Communication Architecture
SDR	Software Defined Radio

TCP/IP	Transmission Control Protocol / Internet Protocol
UMTS	Universal Mobile Telecommunications System
USRP	Universal Software Radio Peripheral

CHAPTER 1

INTRODUCTION

Cognitive radio (CR) is a smart radio transceiver that can sense its environment and adapt itself to available radio parameters, including signal, protocol, operation frequency, and networking. It has the capability of being automatically aware of the environment and adapting their parameters between communication entities. In perfect scenarios, cognitive radio will sense the environment around itself and will seamlessly respond to provide the most benefit to user communication. One of the splendid examples came from J. Mitola, "Cognitive Radio Architecture Evolution,"[1]. Smartphones, when equipped with an ideal cognitive radio, can sense that their owners have introduced themselves to business partners and could then automatically negotiate and exchange their business cards. The phone does not have to be pre-programmed to specific actions, but can adapt itself through case-based reasoning behavior.

Cognitive radio has not advanced to that ideal level yet. Present versions of cognitive radio allow wireless communications to continuously and dynamically operate under limited spectrum, interference environment, or poor channel conditions. Even with current limited capabilities, it still can be applied in multiple real-life applications. For example, an advanced cognitive mobile phone base station can sense and can communicate among others to optimize available channels and invoke self-healing in case of failure. This characteristic increases the availability of the system and reduces the maintenance cost. The IEEE 802.22 [2] standard for Wireless Regional Area Network (WRAN) also uses cognitive radio to utilize the television white space spectrum without

interference. Cognitive radio can also evolve to a more cooperative system, called cognitive network [3]. Cognitive networks use cognitive radio in the physical layer and data link layer, plus additional advanced concepts in network management and expert systems, to create an adaptive network which can automatically adapt itself to the network behavior. This is a similar function shared with cognitive radio but is used in a much larger scale.

To achieve this sophisticated system, a hardware based platform which has the capability to dynamically change itself is needed. Software Defined Radio (SDR) can play a major role in CR development. On the other hand, cognitive radio is an extended version of SDR. SDR is not a novel technology term. It is a radio system implemented by a set of software on a computer instead of a regular deployment using specific hardware, e.g. modulators, multiplexer, A/D, etc. Within this simple definition, software defined radio can be built from many platforms. The first platform might be a personal computer that uses the internal microprocessor as the digital signal processor, the sound card as an analog-to-digital converter, and a homebrew antenna as the radio frequency front end. An alternative possible platform might be a tailor-made design embedded system. Moreover, SDR comes with high flexibility. One single SDR device can receive and transmit multiple radio protocols by altering its software without the need for redesigning the whole platform. A good example of the advantage of SDR might be of a mobile unit which can switch between CDMA2000 and UMTS mobile systems by simply upgrading its software. The concept of SDR will be thoroughly described in Chapter 2.

SDR can create a radio application component known as the “waveform” by connecting virtual hardware modules together through a communication bus. Hardware modules, called “components”, are built using software which modifies hardware virtualizations. The actual hardware that is controlled by a software component is known as the “device”. The combination of devices controlling the entire waveform is called the “platform”. Detailed information will be presented later.

SDR has been introduced with many potential applications in different fields. In military application, especially in the United States, the US armed forces were deployed with different radio system standards and they had a plan to centralize them. With the given number of radio standards used, they were facing a major issue in bridging communication among different armed forces. To overcome such differences, SDR was picked as a novel system which could support multiple protocols by just changing its software. This concept offered a standardized solution for the armed forces to communicate among each other. Additionally, backward compatibility also became available. Another example of SDR application was its use as a disaster recovery communication device. In an example 911 incident, all centralized communication systems failed due to many factors. Base stations were torn apart and existing channels were insufficient. Communication between rescue divisions was not possible because of the lack of a standardized system; therefore, this situation focused great attention to SDR. Similar to the application in the military field, SDR could operate over different standards and made communications between different rescue forces possible. Ad-hoc communications were also made possible because even though existing channels were

entirely occupied, white spaces between mobile units could be used. Furthermore, Cognitive Radio, a more advanced version of SDR, can offer a great solution for spectrum optimization as mentioned above.

SDR consists of two major components, software and hardware. Hardware must be generic and powerful enough to support each radio application. Software has to manage and alter hardware functionality to match the specific application without causing too much overhead. Research communities and radio industries have proposed the concept of unified architecture for both hardware and software and have formed standards among their professional groups. The major purpose of these standards is to achieve software portability and hardware abstraction between vendors and developers.

The professional communities have proposed a rough framework to ensure the capability to perform the required functions. According to these standards, hardware architecture for SDR generally consists of RF front-end, A/D, D/A, digital up-convertor (DUC), digital down-convertor (DDC) and a baseband processor. The same approach has also been applied to software architecture.

Many SDR software architectures have been proposed by various communities, but one of the most widely used came from the US armed forces. The SDR concept has been adopted by the Department of Defense with three main goals: to reduce the development cycle-time, reduce cost, and increase flexibility in communications between branches of the armed forces. To achieve these goals, the Department of Defense has set up the Joint Tactical Radio System (JTRS) program to integrate the use of SDR in military radio systems. JTRS has developed a software architecture framework for SDR

called the Software Communications Architecture (SCA) [4]. It is an open framework with the main objective to facilitate software reuse without hardware compatibility problems. SCA also behaves as a "general manager" in the software level of SDR. SCA acts as a facilitator to load, execute, and allocate application waveforms capacities for each actual device. SCA also manages the communication among components in waveforms and software to hardware through an operating system driver.

SDR was built on the fundamental concept that it had to operate over multiple hardware platforms. Therefore, SCA must also support multiplatform communication. In order to do so, SCA needs an interpreter to translate data communication among different hardware platforms. Hence, JTRS chose a multi-platform communication broker, called the Common Object Request Broker Architecture (CORBA) [5] as a middleware in the SCA. However, SCA has to pay the price for CORBA in CPU cycle requirement, memory consumption, latency, and implementation complexity. Due to CORBA's limitations, many solutions have been proposed to resolve these problems. Such proposals were lightweight structures that eliminate CORBA [6] or new extensions of CORBA [7-9]. However due to many limitations of the multi-hardware standards and backward compatibility, elimination of CORBA might not be an option.

Also in order to recycle software components, developers divided a single waveform into as many multiple components as possible. Doing this reduces workload and development cycle by reusing components with the new waveform. Many articles, which will be discussed later, addressed the fact that creating a number of components

also caused similar overhead problems, CPU consumption, memory consumption, and latency.

As for size and weight of SDR, many factors have to be considered to resolve this issue. Many articles have addressed the problem created because developing more power requires more space for the battery and adds more weight to the system. [10-12]. As mentioned, JTRS has different regulations regarding size, weight, power, and processing power; however, limitations on the battery have not been given enough thought. Since the adoption of SCA, the use of SDR in civilian, aerospace, robotic, and, spacecraft applications, all of which require small-form-factor devices, has become more relevant. Because of the power constraint on such applications [13], SCA design should be aimed at reducing the system's power consumption; and as such, many researchers have focused on reducing power consumption of SDR systems by optimizing the radio's channel-selection algorithms and channel coding.

1.1. Objective and contribution

Balister et al. [14-16] have shown in previous studies that SCA's architecture itself consumes too many resources because of the complexity and diversity it has to support. The tradeoff between component reuse and latency of inter-component communication is one of the primary concerns. Abgrall et al. [17-18] also indicated that the latency caused by the number of components and packet size can be modeled with T-Location distribution, the Generalized Extreme Value distribution, and additional mathematical techniques, even though these techniques were not accurate. In this dissertation, three different methods for choosing SCA parameters were developed.

Specifically, what is presented is a method to choose the optimum number of components, a method to choose the optimum packet size between internal communication, and an algorithm to choose which protocol to transmit in terms of energy consumption.

In chapter 3, an in-depth study was conducted of how the number of components affects latency in the inter-component communications. The results show that the total delay of internal waveform communication does not proportionally increase with the number of components. This inter-component latency becomes saturated after certain number of components has been reached. In these experiments, it became saturated after four components were added to the system. Saturation occurred because latency was overcomed by the routing setup and communication setup. A mathematical model was formed to predict latency by using a logistic function. The model developed in this investigation will allow the design of a better component-reuse scheme. Software Defined Radio designers can use this model to predict the maximum internal latency in terms of number of components, to satisfy communication protocol requirements.

The throughput of SCA is strongly related to the packet size used for data transfer between components. For this reason, there is a need for a method which dynamically identifies the most suitable packet size. This method's requirements should depend on the type of components and number of components used in the waveform. Prediction can be used to form trade-off metrics between component reuse and performance. The metrics will be used to dynamically choose packet size and number of components. In chapter 4, a study of the relationship between packet sizes, number of components, and internal

latency was performed. It was evident that packet delay was increased due to packet size. However, the design objective of this investigation was to observe the relationship between information transmitted per packet and packet delay. The requirement was to find a method to choose the appropriate packet size for SCA waveform. The results have shown that when information per packet increases, the total packet delay also increases but not in a linear fashion. The nonlinearity exists because of overhead caused by the CORBA header itself. Results showed that packet size progressively affects latency when the number of components increases. The outcome of this investigation allows us to design a better model for choosing appropriate internal packet size as a function of the number of components in a waveform.

Originally, SCA was designed to support military applications. In that sense, it had to be designed to support a variety of standards and platforms. Therefore, processing power and battery limitation were not the most important requirements. Subsequently, potential commercial use started to receive more attention. The trend in SDR research has gone more towards commercial civilian applications which require a device with small size, light weight, and low power consumption. Moreover, SDR has a strong functionality in flexibility where a single hardware set can support many waveform protocols. Simultaneously, i.e. one single SDR device can operate three widely used protocols at the same time – WLAN (IEEE 802.11), Bluetooth (IEEE 802.15.1), and ZigBee (IEEE 802.15.4). Each protocol has its own advantages and disadvantages in terms of speed, throughput, and flexibility, but the most concerning issue in this work is energy consumption. SDR possesses the ability to dynamically switch itself between different

radio waveforms. To reduce energy consumption, SDR is required to choose a protocol which consumes the lowest amount of total transmit energy while still maintaining quality of service (QoS). In chapter 5, a comparison of energy consumption between three wireless network protocols was investigated. This comparison showed that even though the low-power protocol device was used, it did not necessarily consume less energy even though, it transmitted the same traffic length. The failure to consume less energy was due to transmitting unnecessary header duplicates. This investigation proposes an energy aware waveform selection algorithm for SDR. The first version of the algorithm developed in this study chooses the lowest energy consumption protocol, depending on the information needed to be transmitted and the switching overhead. Investigation results show this scheme performed better than the use of single protocols specific to each application. The limitation of the investigated scheme was also addressed. When used with non-uniform random distributions of traffic lengths, there were cases where it performed poorer than a single protocol scheme because of consumption of switching overhead. The additional switching condition was resolved by creating a threshold for how many times the protocol had to be selected before an actual switch occurred. Adding a threshold reduced unnecessary switching due to the fact that cases of that information size only appeared once in a while. The average consumption of the proposed scheme in this investigation was better overall, even though, it may not provide the lowest energy consumption in every case. The evaluation of this proposed scheme was conducted through computerized simulations.

CHAPTER 2

SOFTWARE DEFINED RADIO

2.1. What is "Software Defined Radio"

Software Defined Radio (SDR), by first definition according to [19], is "A radio that is substantially defined in software and whose physical layer can be significantly altered through changes to its software".

Back in 1984, E-Systems Inc. (now as Raytheon Intelligence and Information Systems, a division of Raytheon, a major company in aerospace industry and defense industry) demonstrated the software radio concept in their laboratory [20], even though it could not fully operate as software altering radio. After that in 1988, the first public implementation of software defined radio was proposed by Hoeher and Lang at German Aerospace Center for satellite communication [21]. The term SDR had been first used by Mitola III, J. in 1991 and was published as a part of his work [22] in 1992. SDR has several advantages as mentioned especially in military applications. This was highly attractive to the military sector both in United States and Europe so they invested in SDR more in the beginning than the private sector.

2.2. Terminology

In the SDR community, jargons were introduced to define special behaviors and characteristics. Most of them are used differently from their original meanings. This section gives technical explanations:

- **Component** is a stand-alone entity which performs signal processing or control functions. Similar components could be implemented in multiple designs, i.e. two software tunable filter components, both of them perform the similar Low-pass filter operation. One component is implemented by the "Chebyshev filter" but the other one is implemented by the "Butterworth filter".
- **Waveform** is a radio application which is created by just a single component or a combination of components. One major advantages of SDR is the fact that component can be reused over and over with multiple waveforms. For example, an AM radio receiver is a waveform which can be created by combining multiple components--decimator, gain controller, amplitude demodulator, and controller. As shown in Figure 1 from [23], W_1 and W_2 are the abstract block diagrams of waveforms. W_1 consists of four components, $C_{1,1}$ to $C_{1,4}$ but W_2 is a combination of five components, $C_{2,1}$ to $C_{2,5}$. The diagrams also show that the same component may have different designs, like $C_{1,1}$ and $C_{2,1}$ are represented by stacks of blocks.
- **Device** is the actual hardware which performs component functions. Single device can operate using one or more components at the same time. This can include even the whole waveform on a single device.
- **Platform** is a combination of devices which are used to perform the waveform. The most common example is the personal computer which is a single-board platform. A general purpose processor and its sound card are devices of this platform, which is shown as P_1, $D_{1,1}$ to $D_{1,3}$. Figure 1 [23] also shows another platform diagram P_2. This P_2 platform is a combination of two different boards

and could be extended if needed. Examples of the P_2 platform are the SDR development platform from Lyrtech, and Texas Instruments.

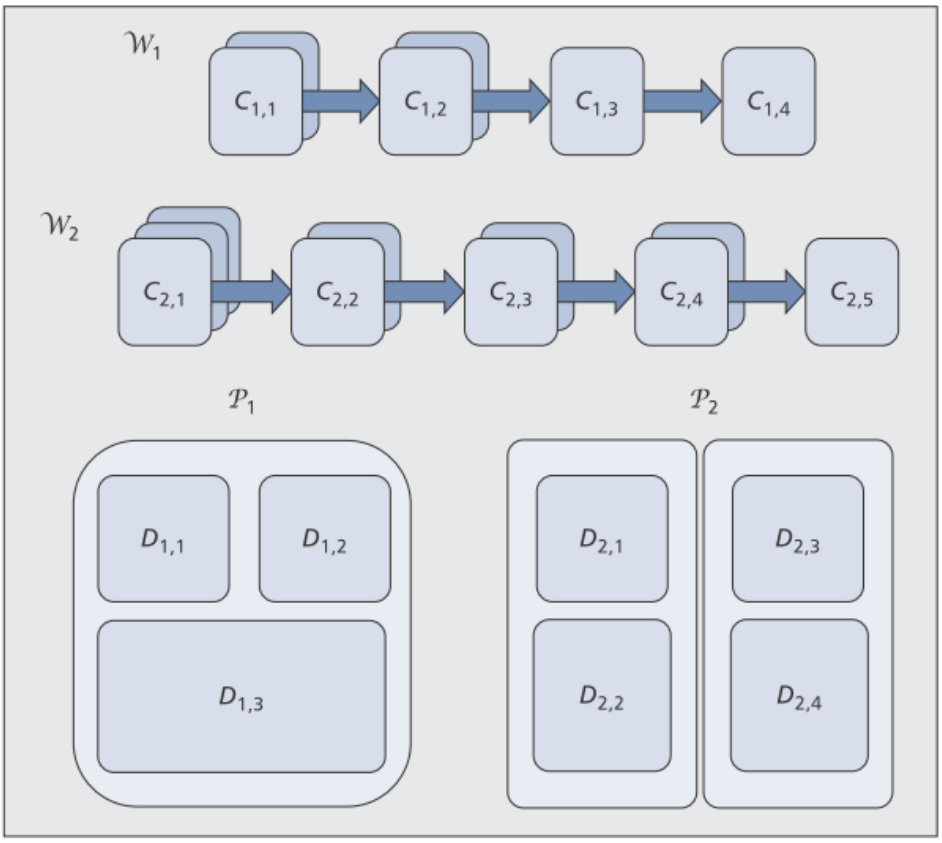

Figure 1: Block diagrams of sample SDR waveforms and platforms [23] © 2009 by IEEE

2.3. SDR Motivation

2.3.1. Multimode functionality

Spectrum sharing is one of the major problems in wireless communication. Spectrum is a limited resource which is always insufficient to share among everyone who needs it. SDR can play a leading role in solving this problem with a practical solution. This is due to the highly flexible characteristic of being a progressively tunable radio

system. A SDR device can dynamically adjust transmission parameters, such as frequency, protocol, range, speed, and transmission power. Example uses of this type of multi-mode radios are:

- Military radios
- Disaster recovery scenarios, since rescue units have a huge gap of communications due to multiple standards. SDR can be a great solution for this problem.
- Smart radio transceiver - Units that can adjust transmission power due to dynamic environment are possible. In many cases, full transmission power causes problems, such as shorter battery life-time, and near-far problems. With SDR, the transmitter can adjust its power depending on the distance between transmitter(s) and receiver(s), size of information, QoS, etc. to improve the whole system.
- Multi-frequency and adaptive directional antenna - Antennas that would help the wireless device to operate between many frequencies at the same time and ignore the interference from unknown sources. This also includes the better range capability because of directional antennas.

2.3.2. High flexibility radio unit

Old school radio units are very limited because all parts are actual hardware implementations. In the past, wireless communication systems and standards were developed rapidly around the world without global synchronization. One of the most obvious issues right now is the mobile phone standard. Different regions around the

world developed their own standards. When people have to travel across regions, it is necessary to change the mobile phone to meet with the local standard. SDR can be a great solution to create a universal mobile device which is capable of supporting multiple standards and having global seamless connectivity [24]. This can also bring the concept of ubiquitous computing to reality.

In the broader range of this issue, Cognitive radio is a novel radio system that can adapt each transceiver in the system to its own operation environment. Each transceiver will sense the environment among each other and then adopt themselves to the most suitable state. This leads to more efficient spectrum utilization [25-27], and advanced security transmission mechanisms.

2.3.3. Reduced cost / Reduced development cycle

SDR is a concept which fully utilizes hardware virtualization. Instead of creating fixed application hardware, SDR uses the concept of hardware virtualization to provide a generic standard architecture. Developers can adapt the software development technique to develop actual hardware. Hardware designers can use this generic hardware to test multiple radio designs by simply upgrading the software without rewiring the hardware. On the software side, software developers can develop a component or a waveform and reuse it over multiple platforms. Also if there is a bug on any component or waveform on a product which is already launched on the market, a vendor can issue software patches over the air (OTA). These process features can reduce enormous cost and time in development cycle. SDR is also an excellent test bench solution in a radio research environment.

Also SDR units which can support multiple radio standards can be produced with fewer discrete components compared to the original radio units.

2.4. SDR Hardware Architecture

SDR Hardware Architecture is a layout of hardware components which has no set hardware system functionality specification. Its components were derived from digital hardwire radio, as shows in Figure 2 [28]. General components consist of:

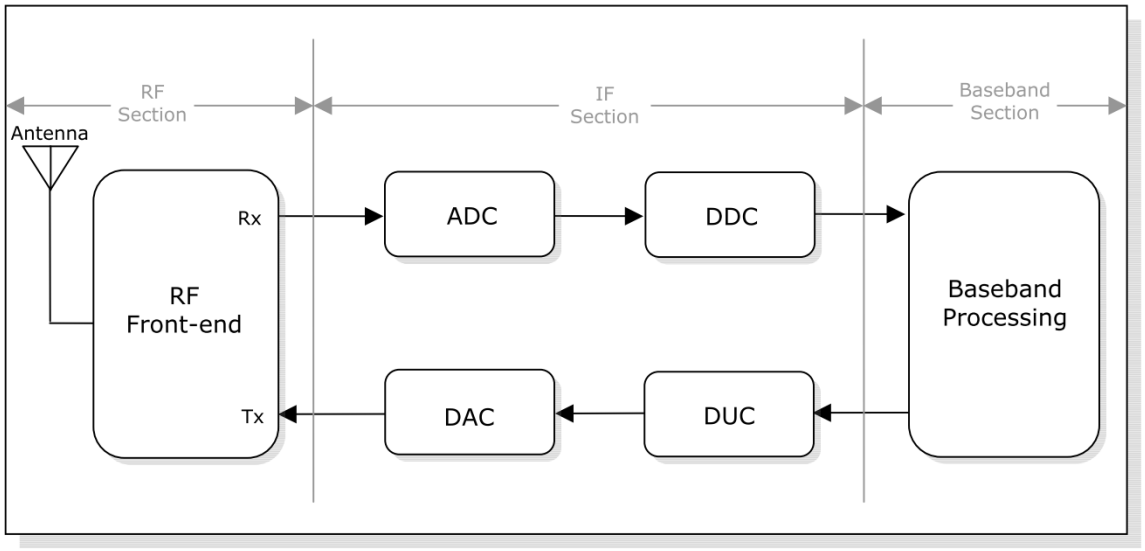

Figure 2: SDR Hardware architecture [28]

- **RF front-end** which is connected to the actual antenna providing conversion between electric current and electromagnetic waves. SDRs have a lot of challenges in this area dealing with creating a configurable

antenna which can operate with multiple directions and frequencies, with low profile characteristics, and with less complexity.

- **Analog-to-Digital converter (ADC) and Digital-to-Analog converter (DAC).** These modules convert analog signal from front-end to digital signal or vice versa. ADC/DAC are the biggest limitations of SDR technology because they have insufficient capability in supporting required bandwidth, range, and sampling rate.
- **Digital up and Digital Down converter (DUC and DDC).** These modules match certain digitized intermediate frequencies to use in follow up basebanded processing, or vice versa.
- **Basebanded Processing.** The actual module performs protocol functions which includes connection setup, equalization, frequency hopping, timing recovery, and correlation. This module is the most configurable part in SDR.

2.5. SDR Software Architecture

SDR needs to have a common operation environment for developers and vendors to follow for implementation of software layers, components and waveforms. This common architecture guarantees that when any waveform is developed, it can be deployed on any platform over and over. The platform independence concept was inherited from the hardware virtualization in software engineering, similar to JAVA

programming on a virtual machine. In order to accommodate this concept, general compositions and functionalities were defined [23]:

- **Application factory.** This constitutes a waveform launcher. It will take care of the initialization process of a waveform on the platform. It is also an information gatherer of each component using loading procedures and connections between them.
- **Capacity Model.** A functionality to profile each platform to determine if they have enough capability to serve the waveform or not.
- **File system.** Used to manage, store, and organize waveforms and components into and from memory.
- **Manager.** A module which performs hardware and software resource management and human user interface.
- **Middleware and Hardware Proxies.** Middleware [29] is a software technique that lets multiple platform computer-like-devices to communicate with each other. SDR needs to have this functionality to support multi-platform communications between non-specific hardware standards.
- **Proxies for physical devices.** SDR must have a communication gateway which can commute with each device in order to pass through data and control information and to configure them.

Example of software radio architectures are Software Communication Architecture (SCA), GNURadio, DttSP, etc.

2.5.1. CORBA

Common Object Request Broker Architecture (CORBA) [30] is an open, non-vendor specific middleware architecture that acts as an interpreter between distributed computer systems, regardless of different system address. CORBA was designed based on the concept of platform independence model (PIM) to ensure that it has multi-platform compatibility. CORBA standard was released by an international organization called the Object Management Group (OMG), where the first version was released in 1991. The formal latest version is 3.2, released in November 2011. Most members of this organization are important companies or consortiums playing major roles in the computer industry such as the Microsoft Corporation, Eclipse Foundation, and W3 Consortium.

CORBA has its own language called "interface definition language" (IDL). IDL is a structure language that provides a common standard for mapping specific languages which need to use CORBA. The examples of languages that can be mapped to CORBA are C, C++, Ruby, Smalltalk, JAVA, COBOL. Applications will generate "generated code classes" by translating IDL to their own languages. These classes will be used to communicate through their internal object adapter. This process is implemented to guarantee that CORBA can communicate among different platforms.

CORBA uses a Client-Server model in communication entities to communicate among CORBA entities. In the abstract level, it is based on the General Inter-ORB Protocol (GIOP). In real implementation, CORBA uses Internet InterORB Protocol (IIOP), an internet protocol version of GIOP. GIOP was mapped to TCP/IP in IIOP so CORBA communications are also based on TCP/IP.

CORBA is widely used among many applications such as banking systems, large scale multiplatform enterprise systems, and embedded computing. This includes a major SDR software architecture, Software Communication Architecture (SCA) [31].

2.5.2. Software Communication Architecture (SCA)

As mentioned before, JTRS has developed one of the most widely used software architecture frameworks for SDR called the SCA. The main purpose of this framework is to facilitate recycling of software components and to ensure compatibility across platforms. In order to open the whole system to multi-vendor design and cooperation, SCA does not have a specific design of hardware and software. JTRS also needs a multi-platform broker to serve communications between multiple platforms which matches the major advantage of CORBA. So JTRS chose CORBA to act as a middleware layer in SCA. SCA was designed mainly to support military applications. As a result, it was designed to support a variety of standards and platforms. To achieve that goal, SCA has been divided into three major components: Core framework, Middleware, and Radio application factory. Core framework takes care of component operations in each device. Middleware takes care of the information transfer between each component. The last is Radio application factory which takes care of the overall waveform operation among many devices.

Figure 3 shows the SCA abstract level diagram [23]. This diagram shows that SCA was designed to work as application manager within the operating system, but in order to serve multiplatform communications, CORBA was needed as an interpreter between each application module. CORBA was placed to work side-by-side with SCA

and provided a communications bus to each application component. The diagram does not specifically show it, but the operating system must also have compatibility with CORBA interface. This means it has to support TCP/IP and CORBA language mapping. An adapter must be built or must be provided for any hardware that does not support CORBA.

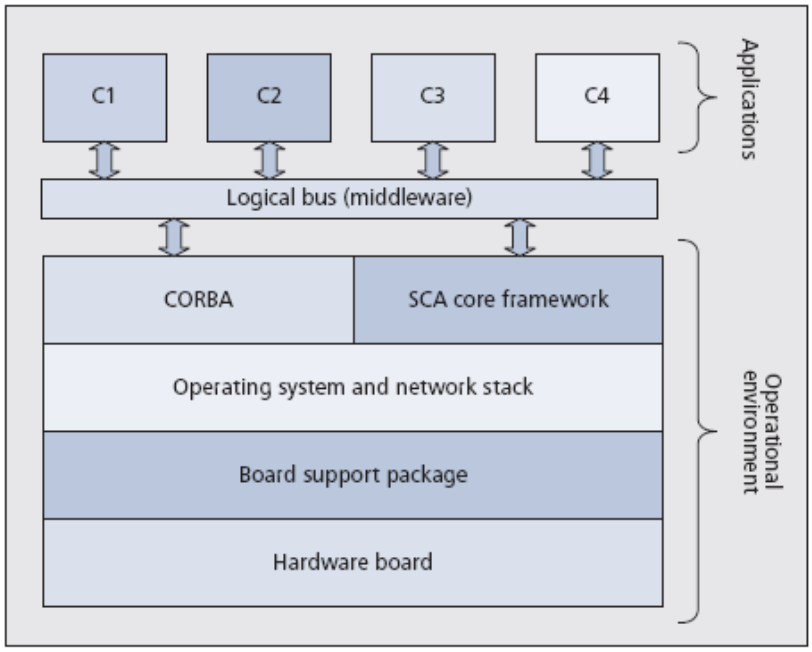

Figure 3: SCA software structure [23]

SCA also needs to meet the PIM standards. To satisfy this, SCA fits itself into the actual hardware communication to the operating system, through the hardware driver. Also SCA requires the operating system to work under the Portable Operating System Interface (POSIX) standard. SCA must have a real time requirement on POSIX in order to provide support for CORBA.

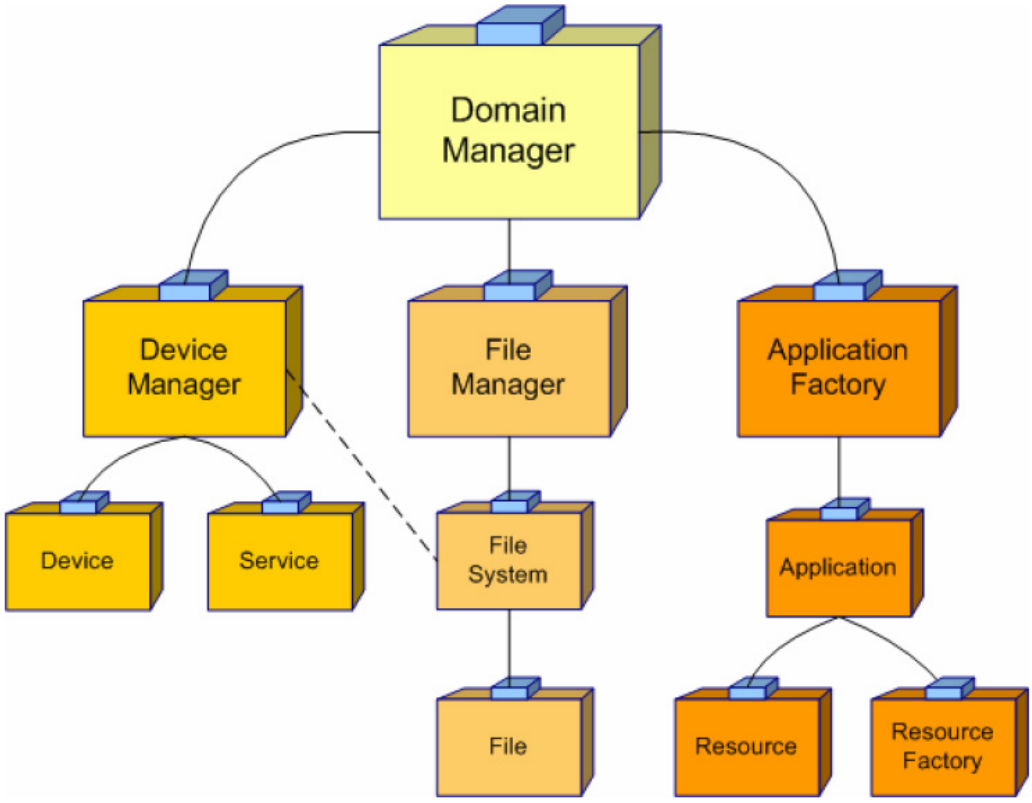

Figure 4: SCA Management Hierarchy at Instantiation [31]

Figure 4 [31] shows the hierarchical software component layout of SCA that is matched to SDR software architecture explained earlier in this chapter. “Domain Manager” is the manager that manages all the hardware devices and software components through each device sub-manager. “Device Manager” acts as a hardware proxy and capacity modeler. This diagram also shows that “Device Manager” co-operates with “File System” to synchronize hardware status to the system. “Application Factory” takes care of waveform launching and interfacing. This includes information management regarding resource allocation and all of these communicate through CORBA middleware

An eXtensible Markup Language (XML) is used in SCA for internal purposes to store hardware capability, properties, inter-dependencies, location of devices and component. XML also has a cross-platform capability which matches the SCA objective.

2.5.2.1. OSSIE

OSSIE [32-33] is an open source SDR which has been developed by Wireless@VirginiaTech group based on the SCA standard. This project has been supported by National Science Foundation (NSF). OSSIE has a main goal to facilitate research communities and SDR education development. OSSIE is used as a teaching tool among many universities so students can understand a SDR and SCA by practice. These teaching materials were co-developed between Wireless@VirginiaTech and Naval Postgraduate School.

OSSIE is operated over a Linux operating system which supports POSIX and TCP/IP by itself. Wireless@VirginiaTech chose to use OmniORB [34] to support CORBA on OSSIE software.

In this dissertation, OSSIE was chosen to use as a based SCA system to investigate the behavior when selected parameters were changed and tested.

2.5.2.2. OmniORB

OmniORB is a CORBA ORB for C++ and Python. It is on GNU GPL opensource License. The original purpose of omniORB was to use on embedded devices at Olivetti Research Ltd, now known as AT&T Laboratories Cambridge. In May 1997, it was firstly publicly distributed under GNU GPL over CORBA communities. OmniORB was

continuously developed in this laboratory until it was closed in 2002. One of the original developer, Duncan Grisby, has kept developing it until today. He formed a Apasphere Ltd company to provide consulting and advising services for the commercial use of OmniORB.

OmniORB is highly compliant with CORBA 2.6 version with some additional functions from the later CORBA version.

2.5.3. TCP/IP

Transmission Control Protocol (TCP) is a protocol implementation corresponding to the transport layer in open systems interconnection (OSI) model. OSI is an abstract architecture standardization of computer device communication systems. Each layer was categorized by its logical function. TCP acts as communicator between an application layer and internet protocol (IP) layer. [35]

In transmitting, TCP serves each application program as a communication gateway. TCP is managed among many applications by using port(s). Each application will have at least one owned specifically owned port. Application(s) communicates through TCP using this port, both transmitting and receiving. Then TCP itself takes care of breaking large chunks of data into packets, attaching it with header information and then forwarding each piece to IP layer. Each piece of information is called “packet”. In receiving, TCP layer receives packets from IP layer, arranges them sequentially based on metadata in their header. The TCP layer acknowledges to the source that the packets have been received. The layer then passes those arranged packets to the proper application by port number referenced in the header. TCP layer also takes care of any error that causes packet loss and duplication. If the packet does not arrive in the estimated time, the

TCP layer source will not receive an acknowledgement and will retransmit that packet again. This process helps the system recover from transmission errors

IP layer is a protocol corresponding to the network layer in OSI model. IP has responsibility in network establishing with two simple functions, “Addressing” and “Routing”. “Addressing” means IP layer is taking care of virtual identity and address of network entity so the other entity can forward the information to the exact entity. The identity is called “IP address”. “Routing” is the function to let each entity to forward the packet to the next entity which is closer to the destination. “Routing” tries to forward the packet with the shortest resource possible to reduce the congestion. Many algorithms have been used in this process such as the “Dijkstra's algorithm” and the “Bellman–Ford algorithm”.

2.6. SCA and Middleware problem

2.6.1. Latency, CPU, and Memory overheads

The evaluation of how CORBA affects SCA was first studied by Balister et al. [14]. In their work, it was claimed that CORBA was the most suitable architecture for SCA based SDR. However, the overhead of CORBA to the SCA was addressed only in terms of processing power. Results showed that CORBA introduces very small processing overhead compared to the baseband processing itself. Murtada et al. [36] also extended the investigation in term of processing power to a specific platform for a better accuracy.

Tsou et al. [15] addressed the CORBA latency issue for SDR even though the CORBA’s latency issue had been addressed earlier but not specifically to SDR. Due to the different characteristics of communication on SDR, latency must be measured using a different method. Tsou’s work compared two SDR systems that have different internal

protocols namely TCP Sockets and Unix Domain Sockets. He showed that the Unix Domain Sockets performed much better in terms of latency. It also addressed the real-time issue of the operating system implemented with FIFO scheduling. In the same year, Balister et al. [16] introduced a method to measure memory consumption of CORBA and SCA. This work showed that CORBA was not the major component that consumed memory compared to the SCA itself.

Abgrall et al. [17], Navarro et al. [37], and Muck et al. [38] compared two different systems, the mono thread non-CORBA SDR (GNU Radio) and the multithread CORBA (OSSIE). It was demonstrated that for both systems, the amount of memory consumption was proportional to the number of components. It was also shown that CPU consumption increased with the number of components; however, this relationship was not linear. Even though Navarro claimed that both systems loaded the CPU and memory equally, Abgrall presented strong evidences that with CORBA, the system consumed much more CPU resources and memory than the non-CORBA system. They showed that only 30% of CPU utilization was used for signal processing. The latency of the system was addressed in terms of packet size. It is obvious that latency will increase with packet size, but it is significant that the system equipped with CORBA introduced much more latency than the non-CORBA equipped one because of the General Inter-ORB Protocol (GIOP) overhead of CORBA.

Abgrall et al. [18] also extended the study of the disadvantages of CORBA related to the performance of SCA in terms of latency. The most important issue was communication between components of the same waveform; CORBA and SCA were

compared and the result showed that CORBA introduced more latency to the system compared to SCA. This latency also varied with the number of components of the waveform. The work introduced and addressed a mathematical technique which can predict latency due to packet size using the statistical model of the T-Location distribution and the Generalized Extreme Value distribution even though they are not absolutely identical.

Although, there are some implementations of real-time CORBA with the SCA standard, all of them are proprietary from the private sector and are custom-tailored designs specific to their own hardware (e.g., PrismTech's e*ORB or Objective Interface's ORBexpress). Even though the real-time CORBA concept has been proposed and implemented from the Object Management Group (OMG) for many other purposes and for a long time, there was no public domain document evaluating the performance and effect of real-time CORBA related to the SCA.

The high potential of civilian applications arose a couple years later and has driven the research community to focus on making SDR more accessible for civilian use. Military designs created many beneficial characteristics with SDR but a number of disadvantages were also included as well. Many challenges in commercializing SDR are still the subject of interest in the software radio research community. Four main issues that have been addressed: performance, size, weight, and power.

2.6.2. Energy Issue with SCA

In terms of power consumption, Dunst et al. [39] introduced a power management solution to a proprietary extended version of SCA. The extension contained a general power-aware computing technique to reduce power consumption plus a new technique for the real-time CORBA implemented by adjusting the GIOP. The work only showed positive experiment results in term of power, but not performance.

CHAPTER 3

STUDIES OF NUMBER OF COMPONENTS VS. LATENCY IN SDR

A part of this chapter was accepted to publish by Inderscience in International Journal of Computational Science and Engineering (IJCSE) [57].

From the discussion in previous chapter, prior studies have showed that the number of components caused a related overhead issue with SDR systems. In compliance with SCA; however, the actual relationship has never been demonstrated. This investigation shows an in-depth study of how the number of components affects internal latency. The objective of this investigation is to indicate the characteristics of internal delay caused by the number of components. These results will lead to better design methods in SDR [40].

This chapter is organized as follows. In section 3.1, experiment methodology is described. In section 3.2, experiment system and environment assumptions are discussed and given. Results and analysis with mathematical modeling are presented and discussed in Section 3.3 and 3.4. In section 3.5, the conclusion is given.

3.1. Experiment Methodology

In this experiment, we observed the relationship between the packet delay and the number of components. In order to do that, we set up a basic waveform consisting of two components, a transmitter and a receiver, as the fundamental system. We varied the number of components between these two entities. Packets were flushed from the transmitter to the receiver through each component. During this process, time delays were measured to observe their behavior.

These experimental waveforms had to be guaranteed that they performed the same application even when the number of component was increased. The component inserted between two basic entities performed nothing except receiving a packet from the previous component then forwarding it to the next component. It also had to introduce overhead as little overhead as possible. To satisfy these criteria, an ideal component called “dummyblock” was created. This approach was also used in prior studies [38, 42] to isolate the overhead. We inserted these “dummyblocks” between the transmitter and the receiver one by one at each round of the experiment. We chose this method because it would not introduce an unexpected delay that does happen with other methods, e.g., split a single component to multiple components.

A time stamp function was added to every component to facilitate measuring time delay between components. As shown in Figure 5, this function stamped a time value when those components completely received a whole packet and completely pushed out a whole packet. The delay between components of each packet could be calculated as a subtraction of timestamp at component N with timestamp at component N-1. The total link delay of each waveform was the summation of these delays.

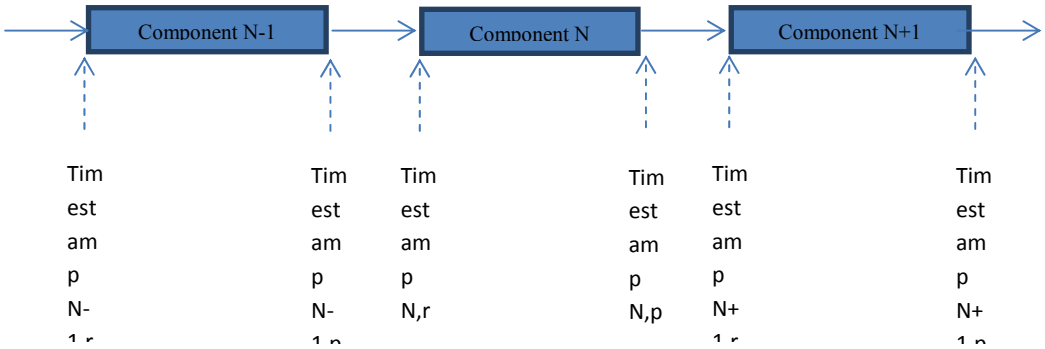

Figure 5: Diagram shows where the timestamps are kept

After calculating the delay between each component, the average delay between each block and average total delay was also calculated for further analysis.

3.2. Experimental system setup and Assumption environment

We performed these experiments on identical custom virtual system. The virtual machine was built on the VMWare Workstation [43] with a single core virtual general purpose 1.7 GHz processor and 1GB delicate RAM. OSSIE 0.8.1 with OmniORB 4.1.4 was running on Linux Ubuntu 10.04LTS. With this system, OSSIE and OmniORB were at the latest version at that time, even though they were updated later.

For each observation, one thousand and twenty four (1024) packets were transmitted from the transmitter to the receiver with five seconds delay between each packet to prevent congestion. Each packet contains one thousand twenty four (1024) bits of information plus the header which was excluded. The experiments were running with nine different waveforms whose number of components varied from two to ten. There was only a single running waveform in each observation. All virtual resources were available as needed.

3.3. Results

Table 1 shows the average time delay of each link between components and the total average delay of each waveform in seconds. The average total delays in each component are between 0.18 msec to 0.6 msec.

Number of Component in waveform	Average time delay between block (Second)									Average total delay between component
	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	
2	0.0018971									0.0018971
3	0.0008492	0.0027311								0.0035803
4	0.0008126	0.0044703	0.0001891							0.0054720
5	0.0011327	0.0039407	0.0002485	0.0002528						0.0055747
6	0.0009602	0.0041932	0.0002099	0.0002100	0.0002246					0.0057979
7	0.0009407	0.0038400	0.0001976	0.0002018	0.0002063	0.0003140				0.0057005
8	0.0008442	0.0036638	0.0002124	0.0002182	0.0002288	0.0002240	0.0002124			0.0056038
9	0.0008853	0.0035944	0.0002000	0.0001991	0.0002135	0.0002378	0.0002429	0.0002187		0.0057917
10	0.0008928	0.0031795	0.0002331	0.0002232	0.0002298	0.0002445	0.0002594	0.0002150	0.0001987	0.0056761

Table 1: Table of average time delay vs. number of component

3.4. Analysis and Modeling

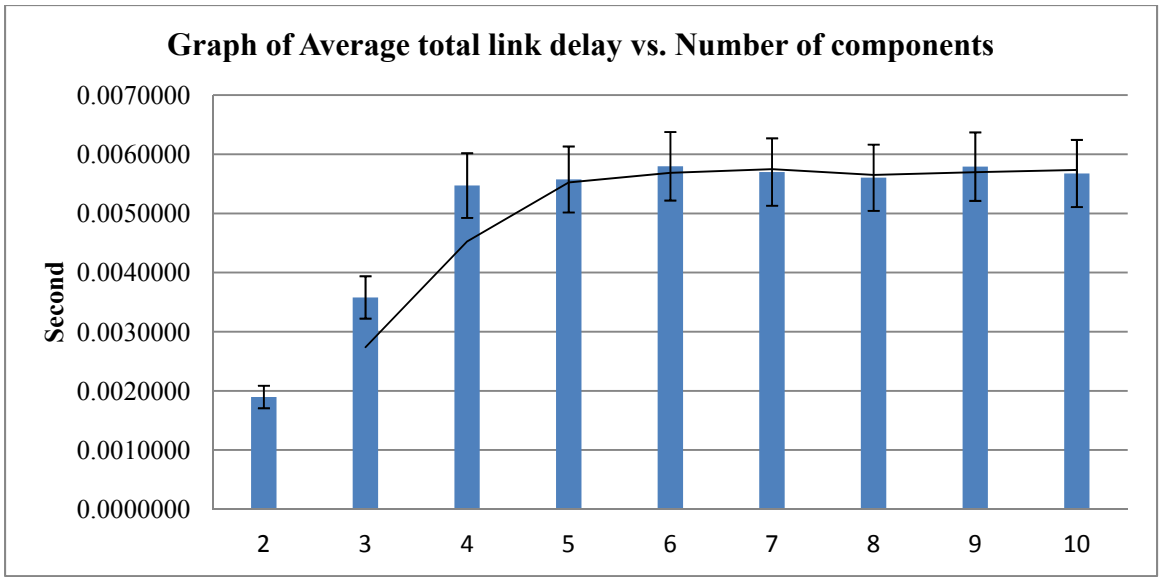

Figure 6: Graph of Average total link delay vs. Number of components

The results show that the total delay did not increase proportionally with the number of components. The total delay saturated after a certain number of components is reached. As shown in Table 1, the total delays increased due to the increasing number of components in the beginning. However after we inserted the 4th dummyblock component, the total delays started to saturate. The result is plotted in Figure 6 as a graph of the total delay versus the number of components. This plot clearly showed the saturation.

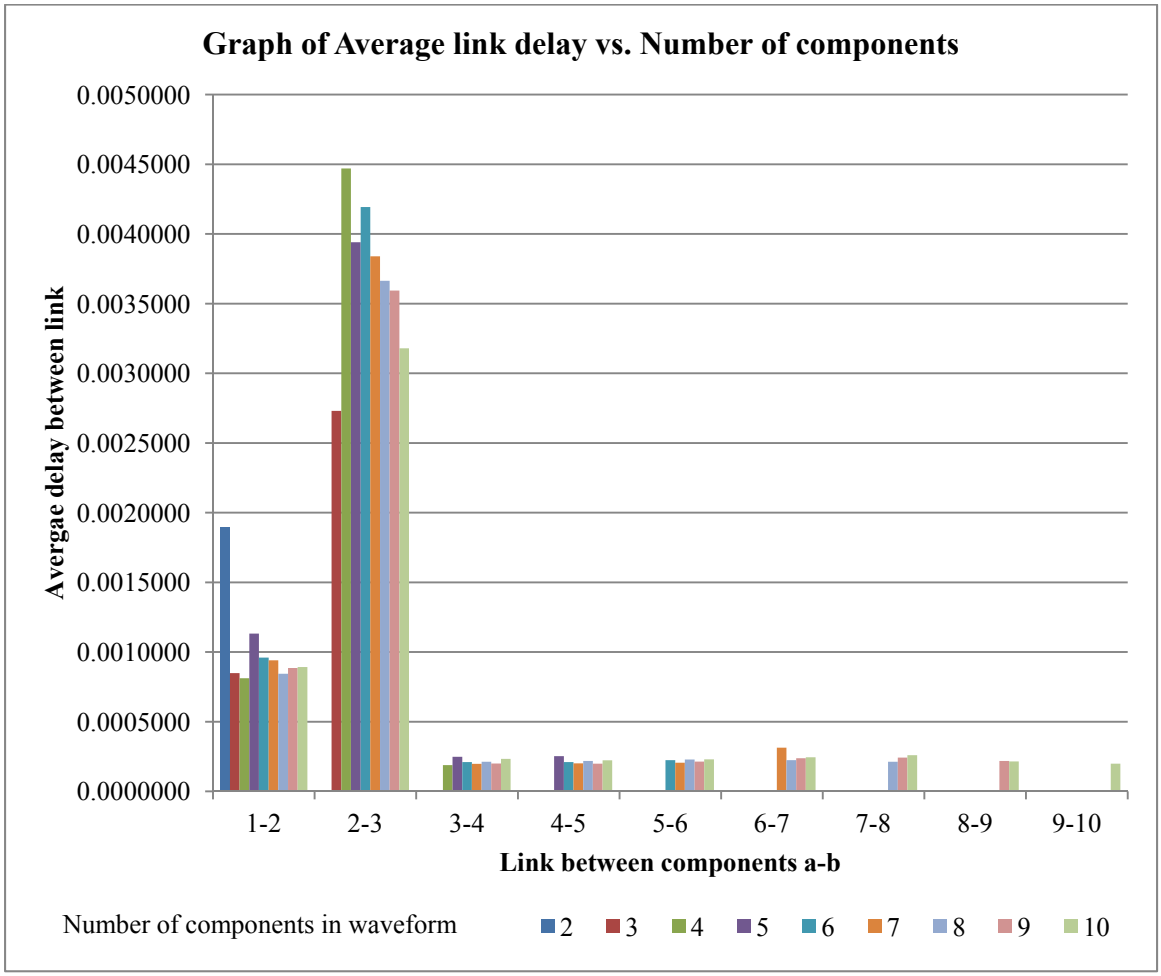

Figure 7: Graph of Average link delay vs. Number of components

Also, it was observed that the average transmission time delays between dummyblock component 1-2 and component 2-3 were dramatically high compared to the others. This occurred because the majority of time delays came from communication setup. Figure 7 clearly illustrated this observation. The majority of time delays of each waveform were due mainly to the delays between the 2nd dummyblock and the 3rd dummyblock. It also significantly indicated that time delays between the 4th dummyblock to the 10th dummyblock were equally distributed. Only time delays occurring between the 2nd dummyblock and the 3rd dummyblock increased when the number of components was increased.

CORBA was taking care of the communication between each node in the SCA network. CORBA was based on a GIOP client-server model which caused CORBA to initialize communication as an IP based system. The setup process is composed of:

- Setting up routing table.
- 3-way handshake setup.
- CORBA Communication setup

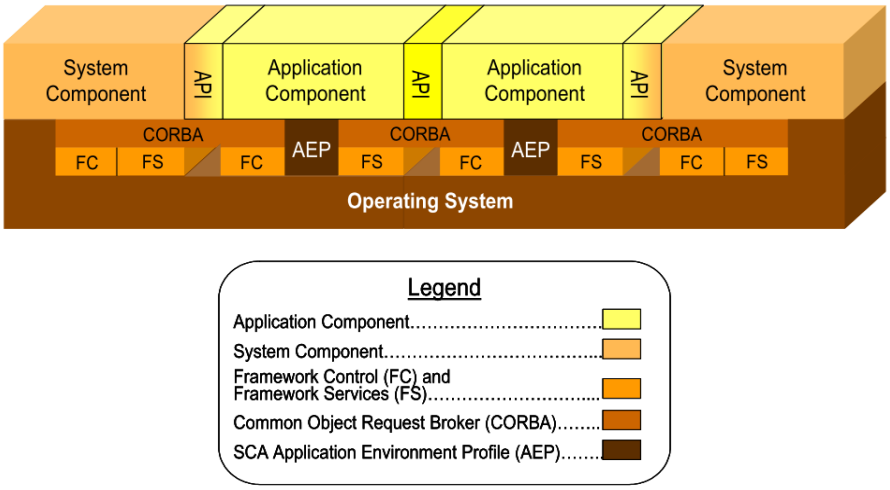

Figure 8: Generic representation of abstract level waveform communication [31]

Also, another delay issue that should be addressed is from the SCA structure. In the abstract level, components are virtualized so that they can communicate among each other directly without any limitation, like in Figure 8 from SCA specification [31]. In practice, component communication performs quite differently from the abstract level illustration. Components cannot directly commute among each other. Signals, or data, must be forwarded through a centralized bus system. Various SDR architectures handle this bus with middleware, especially in SCA. Middleware supports each component by acting as an interpreter. Middleware is absolutely necessary due to incompatibility among hardware I/O and multi standard communication protocols. Figure 9 illustrates a better representation of how components communicate among themselves. These components have to share a single centralized bus communication among them.

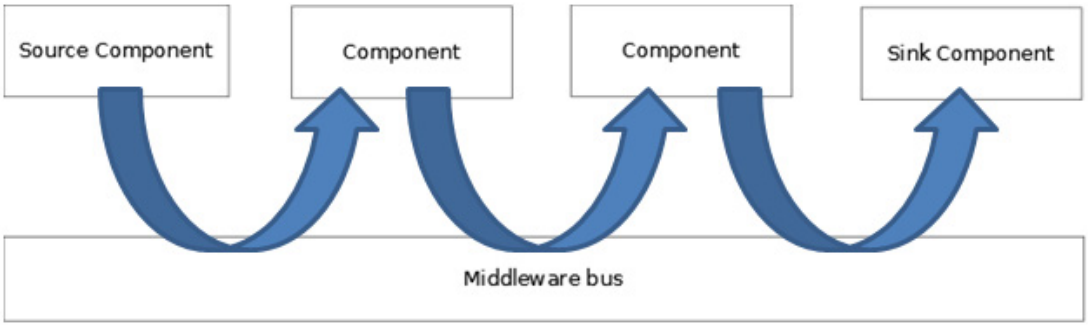

Figure 9: Generic representation of actual internal waveform communication

This technique, obviously, causes problems in term of performance to those SDR architectures as shown by the experiment. If the actual physical bus has not been well designed, this will increase the time delay, especially when components share the communication bus.

The prior studies [17, 38] showed the link with delay overhead related to SDR systems. Results in this investigation support the prior studies. The results show similar trend and values with delays even though experimental environments were not the same. Prior studies gave more attention to the CPU and memory consumption overheads than to the delays in term of the number of components. The prior studies showed only delay of single link between components which was not the representative of the overall system. The delay was between 120-400 microseconds which were similar to that of this investigation. The similar approach [38] in isolating the processing delay and overhead from the measurement was used. This current investigation measured delay more in detail to focus especially in term of the number of components. This investigation also showed

similar saturation of the delay when the number of components reached some certain number. Even though SCA-CORBA was not used, the investigative environment of this study was based on a centralized manager. Additionally, prior studies had never proposed any mathematical model relating the delay behavior in term of the number of components.

In order to analyze the behavior of the delay due to the number of components, the CSMA/CD mathematical model was adopted. CSMA/CD model [44-45] was picked initially because both of these models shared a similar topology; multiple entities are sharing the same communication bus.

Giving the assumption, N components contend to transmit through the middleware bus, which is similar to the share channel in CSMA/CD. The probability that some component would successfully transmit is:

$$P_{success}=n\cdot p\cdot(1-p)^{n-1} \tag{3.1}$$

This can be maximized by choosing:

$$P_{optimal}=\frac{1}{n}\rightarrow \max(P_{success})_{P_{optimal}\text{ and }n\rightarrow\infty}=\frac{1}{e} \tag{3.2}$$

So the average number of time slots until some component successfully allocates the bus will be:

$$E[x]=\frac{1}{P_{success}}=e=2.718 \tag{3.3}$$

Therefore the average contention interval is given by

$$\text{Average contention interval} = e \cdot 2t_{prop} \tag{3.4}$$

And the average time delay is:

$$D = X + e \cdot 2t_{prop} \approx x + 5t_{prop} \tag{3.5}$$

where

X = Packet transmission time

This is an attractive model to describe the behavior of the delay due to the number of components, however CSMA/CD communication is just similar to but not the same as the inter-component communication topology. In CSMA/CD, all entities will compete with each other to use the channel, however, as shown in Figure 9, in SDR inter-component communication is a cascade system where the packet will be transmitted from entity 1 to entity 2, entity 2 to entity 3, entity 3 to entity 4, etc. The CSMA/CD adopted model may not perfectly reflects the behavior of the inter-component communication.

The trend line in Figure 6 clearly shows the saturation of this delay. The behavior of these delays is similar to the famous S-shaped function, "Logistic function" [46-47]. The logistic function is characterized by an approximately exponential growth rate in the beginning followed by saturation. The logistic function is used for modeling in many fields such as demography (the population growth model), medical (the growth of

tumors), and physics (the Fermi distribution). The simple logistic function can be described as

$$f(x)= \frac{1}{1+e^{-x}} \tag{3.6}$$

which can be generalized as

$$Y(t)=A+ \frac{K-A}{1+Qe^{-r(t-t_0)}} \tag{3.7}$$

where

Y is a function of t
A : the lower asymptote;
K : the upper asymptote.
If A=0 then K is called the carrying capacity;
r : the growth rate;
Q : depends on the value Y(0)
t_0 : the time of maximum growth

To find a closed form model of this delay, the values of parameters listed above need to be adjusted. This method is known as parameter extraction. In the beginning the actual data between total delay and number of components was plotted into the x-y plane, as in Figure 10, to estimate the growth rate and time of maximum growth. The lowest possible delay is 0 therefore the lower asymptote is 0, which leads to the value of one for Q.

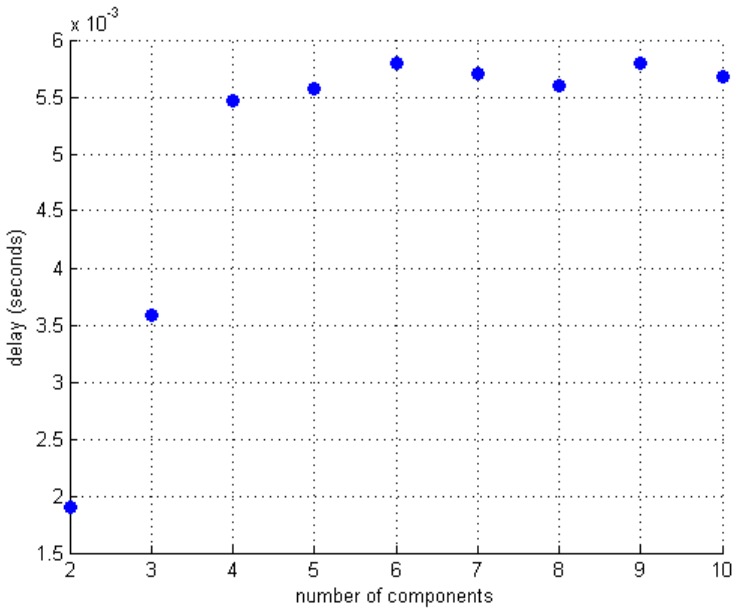

Figure 10: Graph of Actual average total link delay vs. Number of components

The estimation range of possible growth rate (r) was between 0.01 and 4 and the possible time of maximum growth (t_0) was between 0.1 and 5. These estimation ranges were used to solve for the upper bound (K). To solve for K, the least square method was used to find the best fit spots of r and t_0. The objective of this method is to minimize the distance between the function and actual delays.

$$\min_{r,\, t_0} \left\| h - M \right\|^2 \tag{3.8}$$

where

$$h = \frac{1}{(1 + e^{-r(t - t_0)})} \tag{3.9}$$

M = actual data

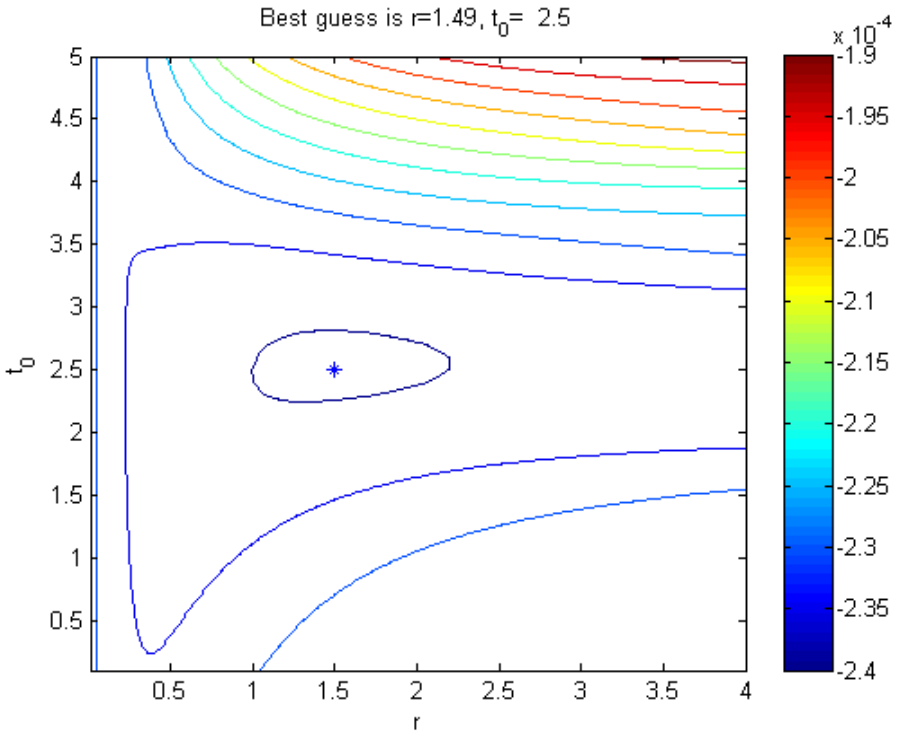

Figure 11: Contour plot of least square method

In Figure 11, the best fitting point was marked as "*" in a contour plot of least square method. The best fitting values were $r = 1.4987$ and $t_0 = 2.5403$ which gave value of $K = 0.005746$. The closed form expression that approximated best the data point is given by the following expression.

$$D(t)=\frac{0.005746}{(1+e^{-1.4987(t\ -\ 2.5403)})} \tag{3.10}$$

where

K : the upper bound of the delay.
r : the growth rate of delay, in exponential region.
t_0 : the number of components which gave the maximum growth.

The closed form expression in e.q. 3.10 was plotted with the actual data in Figure 12.

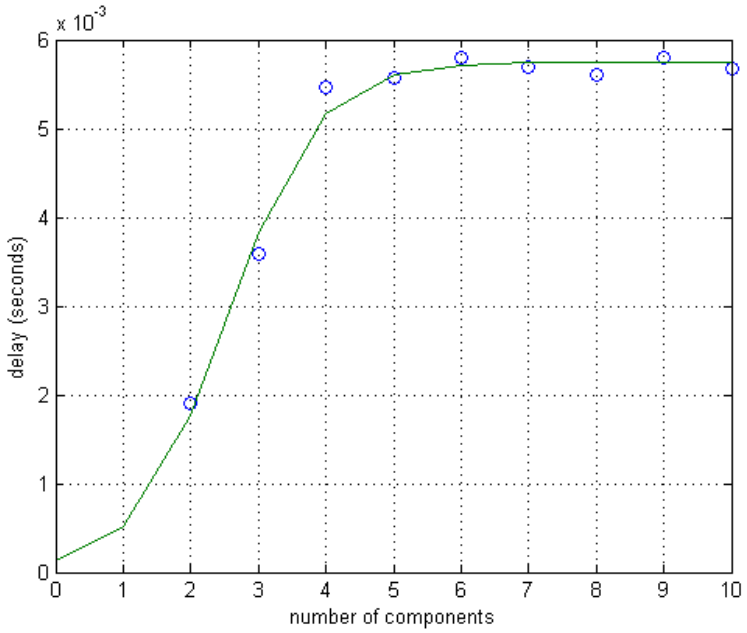

Figure 12: Graph of average total link delay vs. Number of components

In Figure 13, the absolute errors between actual delays and the closed form expression were plotted. Figure 14 showed the relative error between them. The plot presented the goodness of fit of this model which performed very well when the number of components was large. The goodness of fit also addressed higher error when the number of components was small. A maximum percentage error was 6.73%.

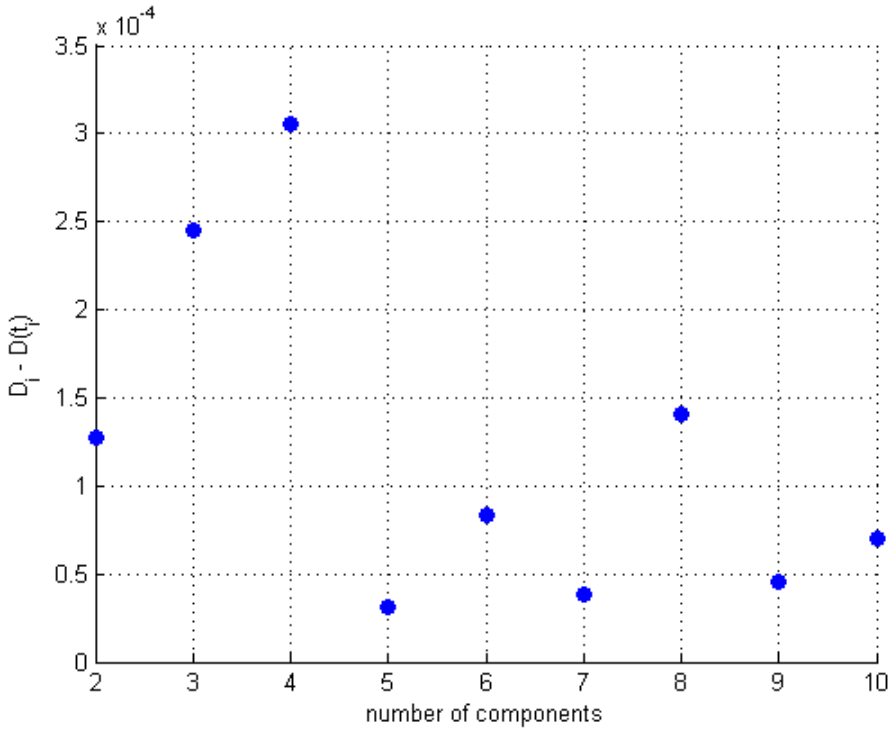

Figure 13: Graph of Absolute error vs. Number of components

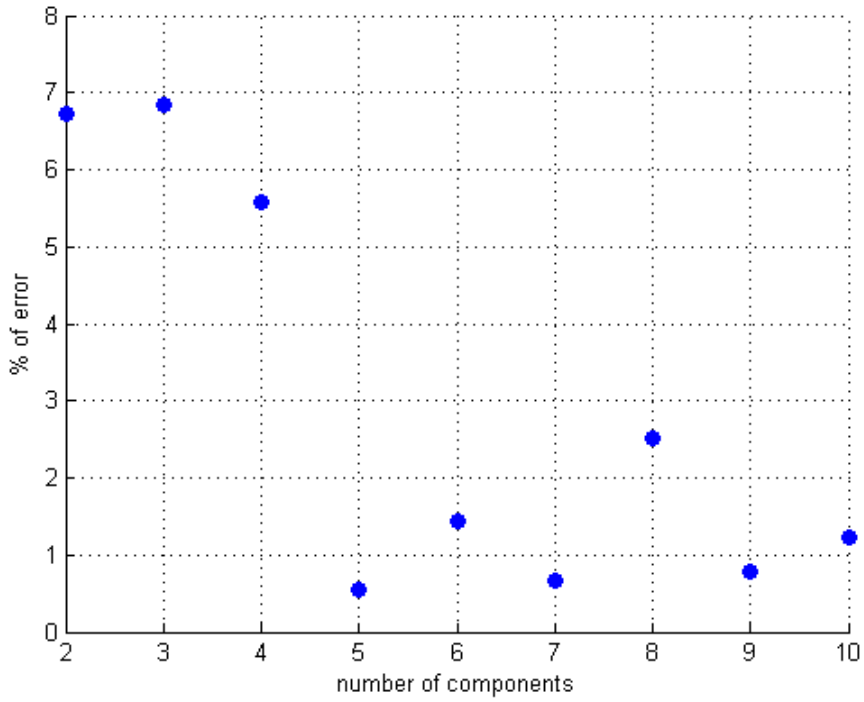

Figure 14: Graph of relative error in percentage (%) vs. Number of components

The experiment was conducted again with the five new waveforms to test the validity of this model. One dummyblock was added incrementally from eleven to fifteen dummyblocks with each waveform. The closed form expression in e.q. 3.10 was plotted with the actual data in Figure 15. In Figure 16, the absolute errors between actual delays and the closed form expression were calculated and plotted. Figure 17 showed the relative error between them. The model performed very well compared with independent observed data. A maximum percentage error was 0.2-1.4% when the number of dummyblocks was increased to eleven to fifteen incrementally.

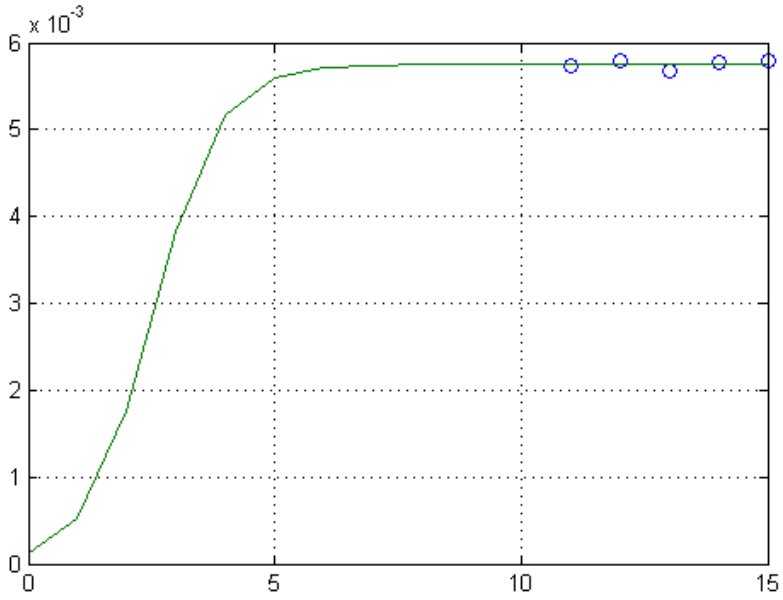

Figure 15: Graph of average total link delay vs. Number of components

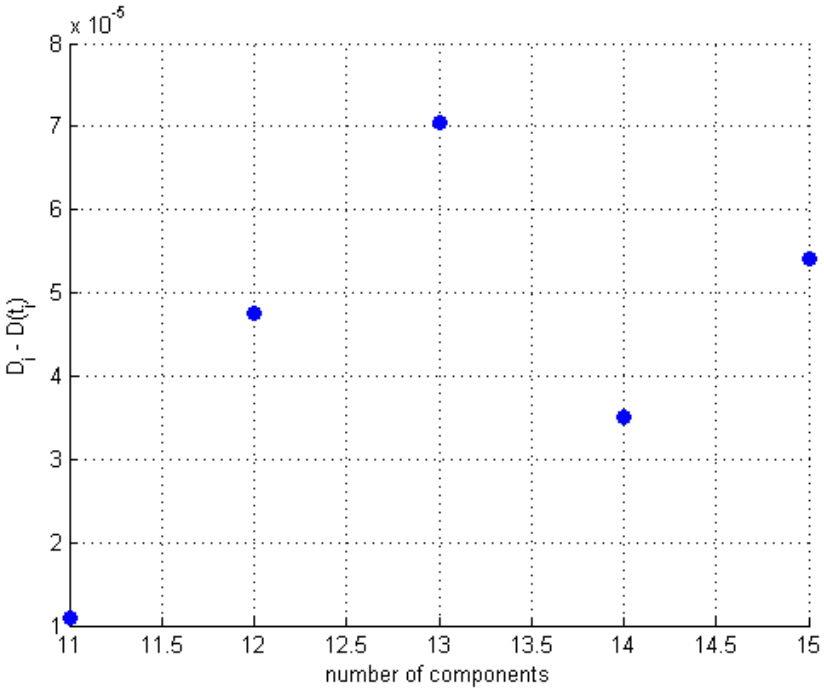

Figure 16: Graph of Absolute error vs. Number of components

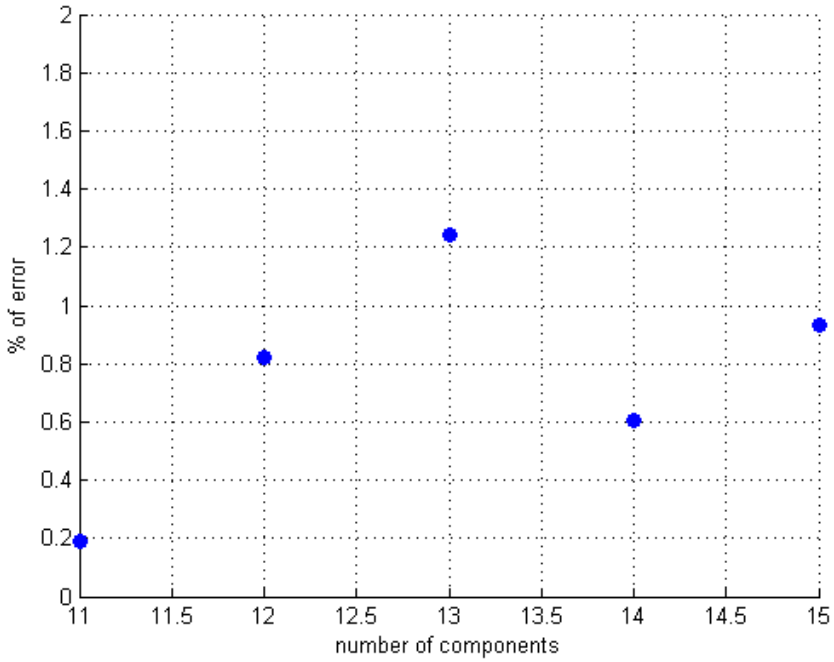

Figure 17: Graph of relative error in percentage (%) vs. Number of components

This closed form expression of the delay versus number of components can be used for designing better SDR waveforms. The closed form expression can be used in predicting the link delay related to number of components in the SCA-SDR system. The model can help the software designer to match the number of components planned to be included in the waveform design to satisfy the protocol limitation. The model can also give a more realistic delay in the simulation which leads to more practical results.

3.5. Conclusion

In this chapter, a study of the internal inter-component communication delay due to number of components was presented. Through controlled environment experiments, it was demonstrated that the total link delay will increase due to the number of components

in the beginning. The delay will saturate after a certain number of components in the waveform is reached. A closed form expression is obtained by using a logistic function to model the behavior of the delay as a function of the number of components. The accuracy of this model was presented by comparing it with actual data. The model had less accuracy when the number of components was small, however it was better when the number of components was large.

CHAPTER 4

STUDIES ON INTER COMPONENT COMMUNICATION LATENCY BASED ON VARIATION OF NUMBER OF COMPONENTS AND PACKET SIZE IN SDR

A part of this chapter was accepted to publish by Inderscience in International Journal of Computational Science and Engineering (IJCSE) [57].

In the previous chapter, experiments were done by varying the number of components to observe its effect on delay. However Abgrall et al. [17-18] also indicated through some evidence that internal packet size did affect the average internal delay. In this current investigation, we extend the experiment to identify the effect of packet size on the delay. This investigation also considered a relationship between the number of components and packet size. The result of this experiment will lead to a better approach in designing communication criteria.

This chapter is organized as follows: In section 4.1, experiment methodology is described. In section 4.2, experiment system and environment assumptions are reviewed and given. Results and analysis with mathematical modeling are presented and discussed in Section 4.3 and 4.4. In section 4.5, the conclusion is given.

4.1. Experiment Methodology

It is clear that delay increases with packet size. However the objective of this investigation is to observe the relation between information transmitted for each packet and the resulting packet delay. The main goal is to find a better approach for choosing packet size utilized in internal SCA communication.

In this investigation, the average delay was measured. The delay was one which occurred between internal waveform communications resulting from different packet

sizes. Multiple packet sizes were tested with 9 different waveforms used in the previous chapter investigation (Section 3.2). The difference between each waveform was the number of components. The number of components varied from two to ten. The fundamental waveform and dummyblock from previous chapter were used to prevent the unexpected delay.

The packet size was changed by adjusting transmission parameter in a CORBA manager.

4.2. Experimental setup and Environment assumption

We still performed these experiments on the same custom virtual system as the previous work. The virtual machine was built using VMWare Workstation featuring single core general purpose virtual 1.7 GHz processors with 1GB delicate RAM. OSSIE 0.8.1 with OmniORB 4.1.4 was run on Linux Ubuntu 10.04LTS in this system.

Packet sizes were varied in each experiment but the total transmission information was kept constant at 1 Megabits. The packet sizes were 256, 512, 1024, and 2048 bits, so the total number of packets were inversely related with 4096, 2048, 1024, and 512 packets respectively. Each experiment was run with nine different waveforms with the number of components varied from two to ten, similar to the experiment in previous chapter. Packets were transmitted from the transmitter to the receiver with five seconds delay between each packet to prevent congestion. There was only a single version of waveform running in each observation. All virtual resources were available as needed.

4.3. Result

In Figure 18 to Figure 26, the results for each number of components are plotted as the average of total link delay versus packet size. The results clearly show a result as expected that the average total link delay also increased when the packet sizes increased. Results also showed that when the number of components was small, the delay did not increase as expected.

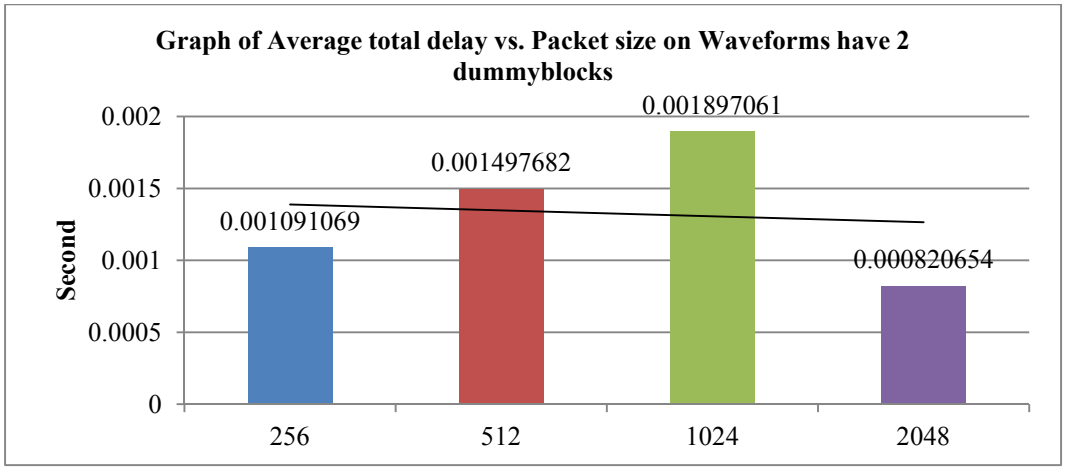

Figure 18: Graph of Average total delay vs. Packet size. The waveform has 2 dummyblock components.

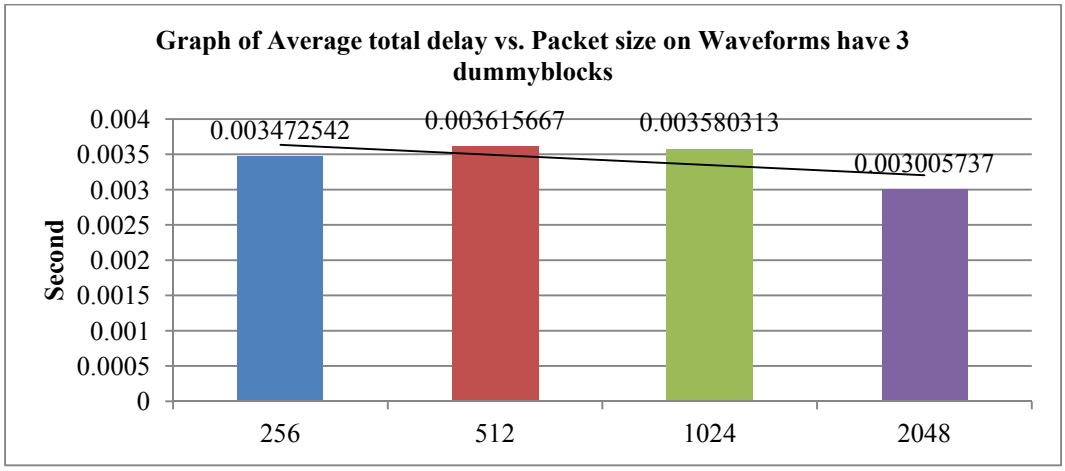

Figure 19: Graph of Average total delay vs. Packet size. The waveform has 3 dummyblock components.

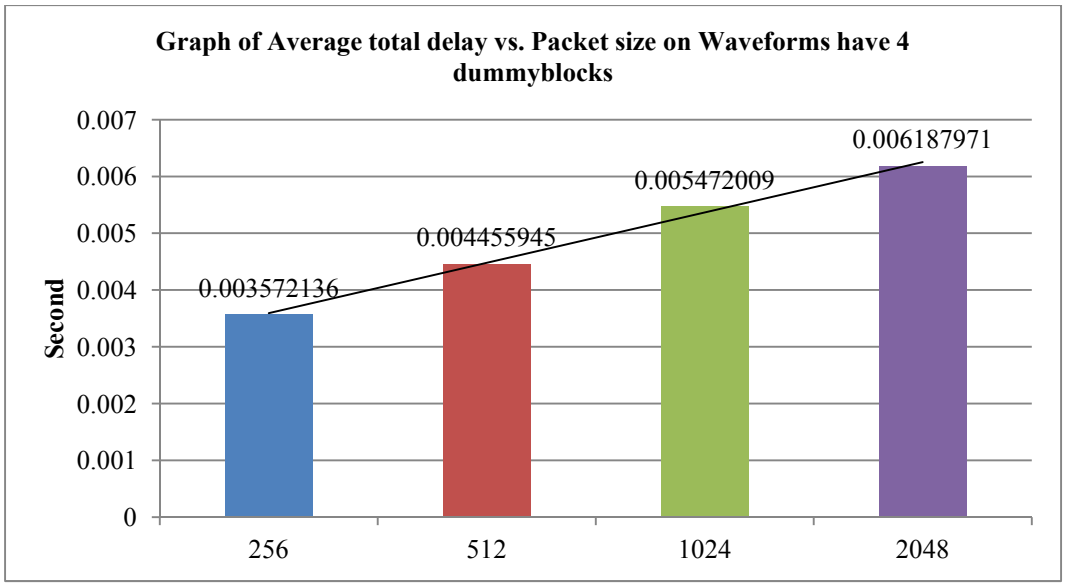

Figure 20: Graph of Average total delay vs. Packet size. The waveform has 4 dummyblock components.

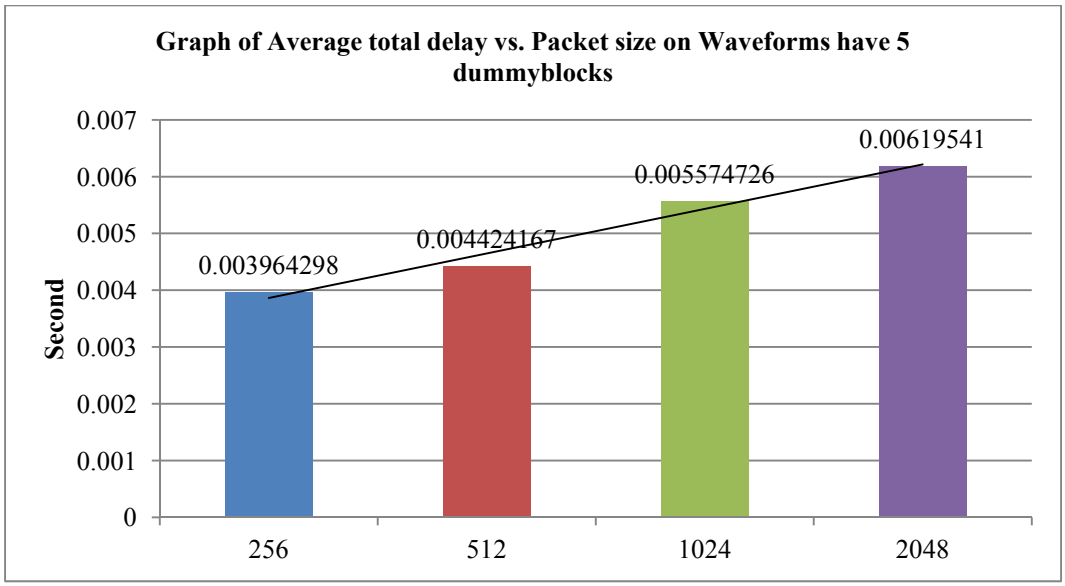

Figure 21: Graph of Average total delay vs. Packet size. The waveform has 5 dummyblock components.

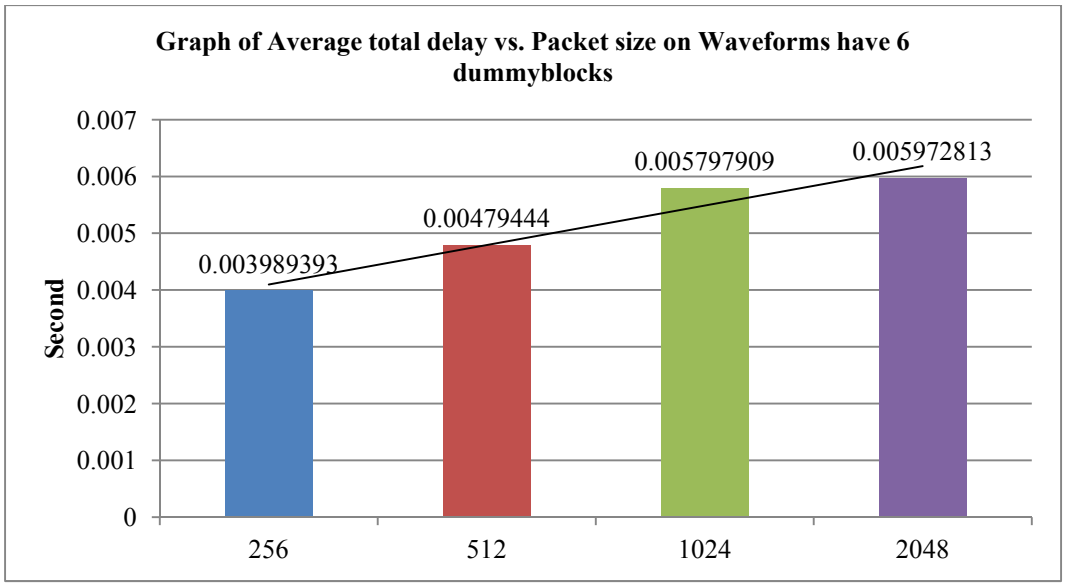

Figure 22: Graph of Average total delay vs. Packet size. The waveform has 6 dummyblock components.

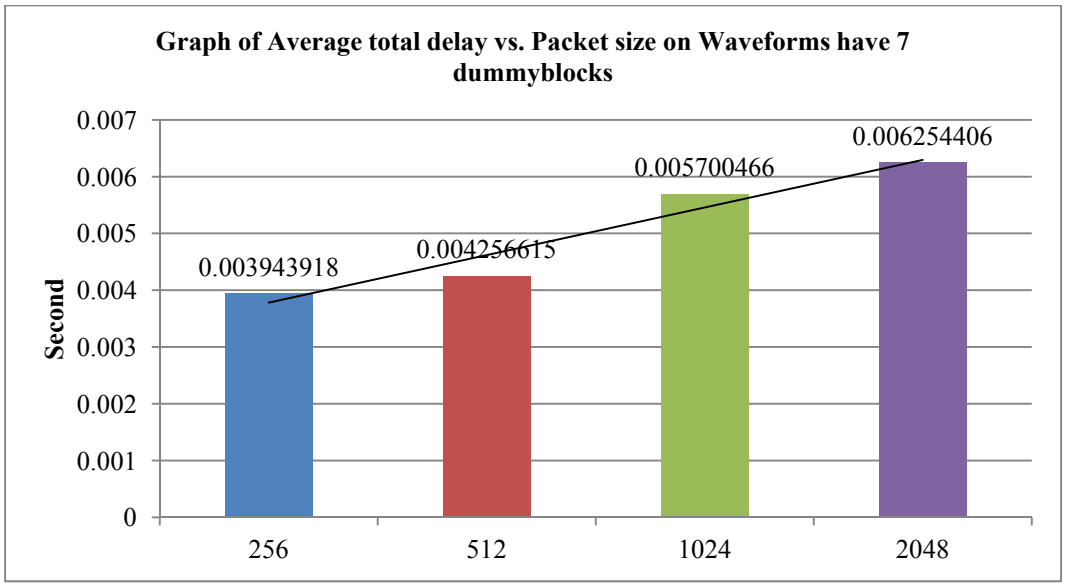

Figure 23: Graph of Average total delay vs. Packet size. The waveform has 7 dummyblock components.

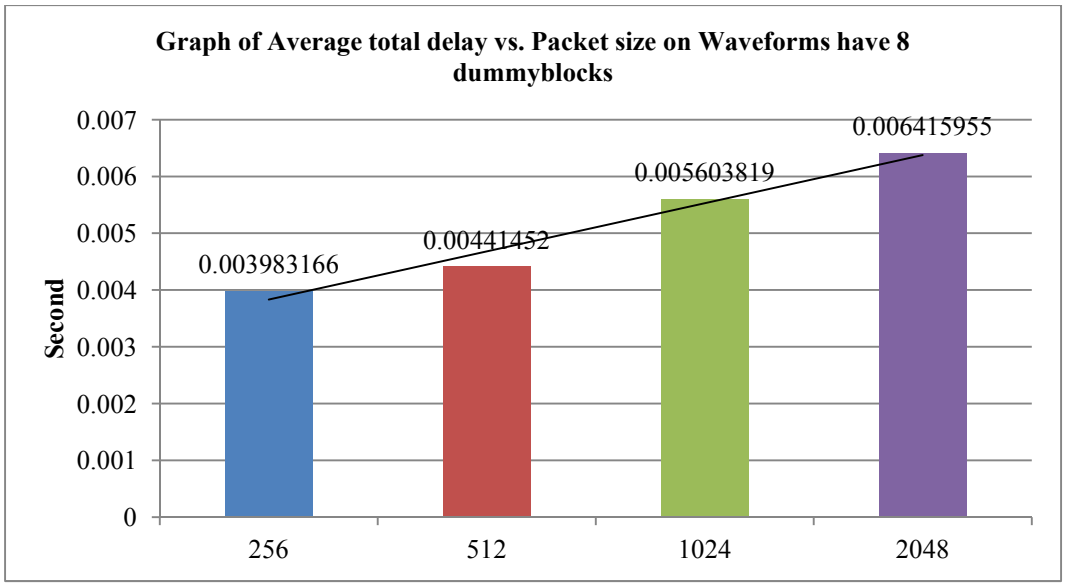

Figure 24: Graph of Average total delay vs. Packet size. The waveform has 8 dummyblock components.

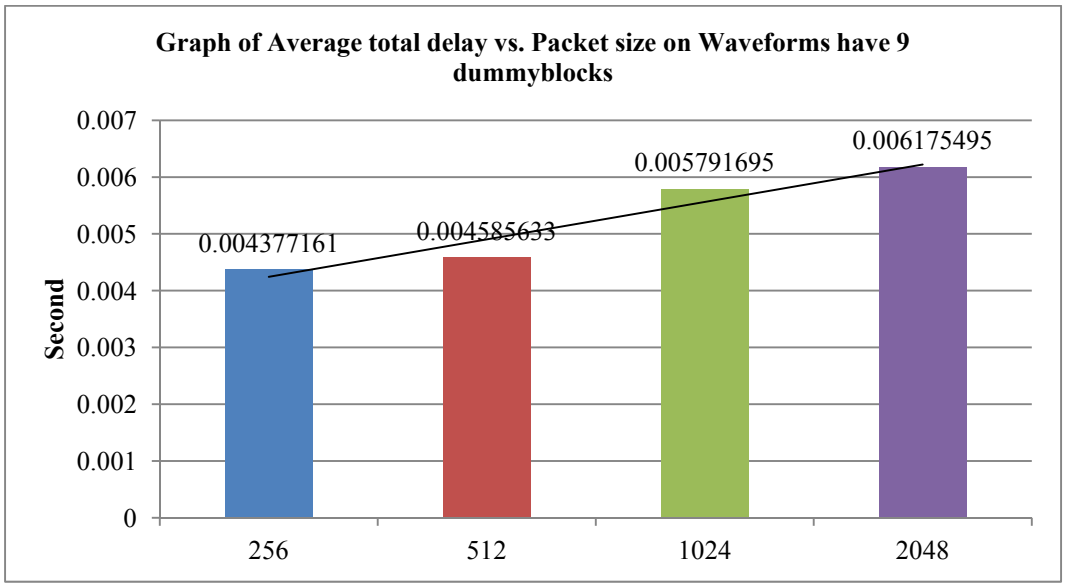

Figure 25: Graph of Average total delay vs. Packet size. The waveform has 9 dummyblock components.

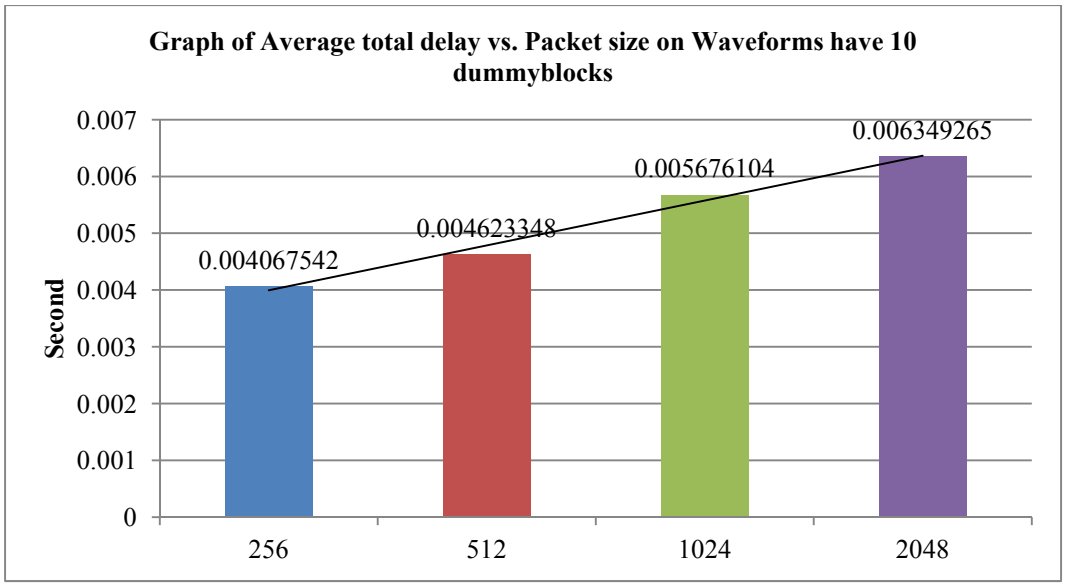

Figure 26: Graph of Average total delay vs. Packet size. The waveform has 10 dummyblock components.

4.4. Analysis and Modeling

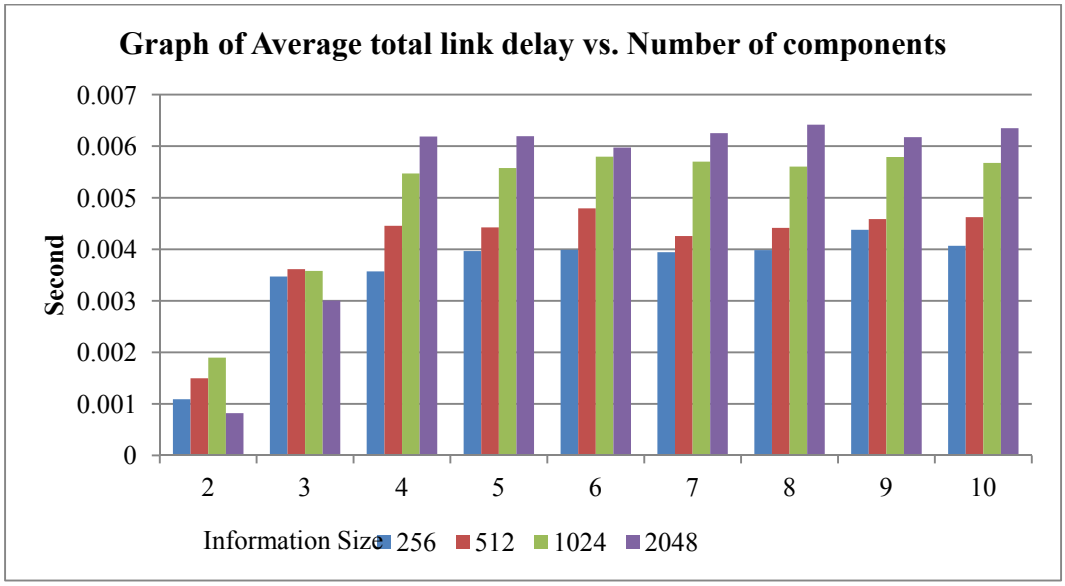

Figure 27: Graph of Average total link delay vs. Number of components for four different packet sizes

Figure 27 shows the combined results of average total link delay versus number of components on four different packet sizes. Even though the delay increased with packet sizes, it was not linear. The growth rate of delay was in logarithmic function due to the packet size. It was observed that the packet size has less effect to the delay when the number of components is small. This effect is explained by the fact that the actual transition delay was much less than the delay from the packet processing overhead generated from CORBA manager itself [14-16, 36-37]. The CORBA manager overcomes the total link delay.

In previous studies, Tsou el al. [15] had a different experimental environment from our investigation. They compared two different SDR system, CORBA and non-CORBA. Their results showed similar delay at some packet sizes. Their results also showed the delay in term of packet size in CORBA system was a step function. Their experiment did not isolate processing delay from the components. Abgrall et al. [17] measured the delay in term of packet size only at single link between components even though their waveforms consisted of multiple links between components. Their packet sizes concerned only actual information, but did not include header and encoding overhead, as this investigation did. The current investigation results support their findings where trends and latency values with similar data sizes were used provided the packets did not have communication set up delay. This investigation was more extensive by measuring the delay more in detail. These details showed that the delay was overcome by communication setup.

Abgrall et al. [18] extended their study again later with by measure parallel links and purposed a statistic model using a T-location Scale distribution and a Generalized Extreme Value distribution to fit the delay. Prior studies had never proposed any mathematical model to explain the delay behavior in terms of the number of components and packet size or any consideration of the relationship between them.

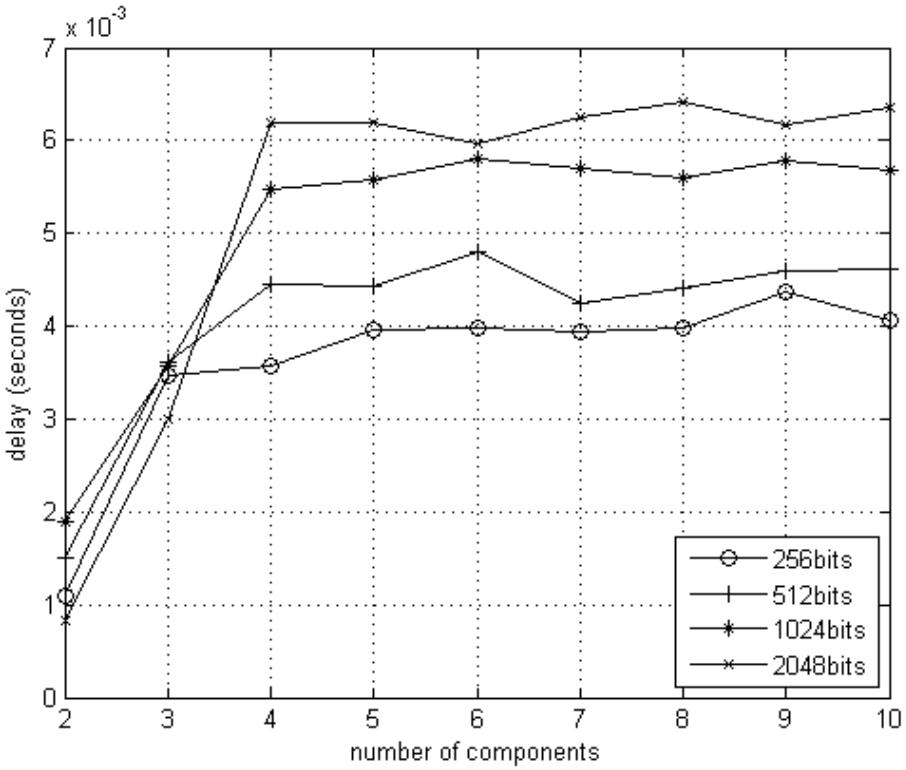

Figure 28: Graph of Average total link delay vs. Number of components for four different packet sizes

In Figure 28, the delay of each packet size is shown in a similar trend. After a certain number of components had been reached, the delay became saturated. From the previous chapter, this was modeled with the logistic function.

$$Y(t)=A+\frac{K-A}{1+Qe^{-r(t-t_0)}} \tag{4.1}$$

The parameter extraction method was used again, similar to the previous chapter with the four different packet sizes creating each individual model. Table 2 presents those parameters. It is observed that upper bound of the delay (K) and maximum growth rate (t_0) increased as a logarithmic function of packet size.

	Packet sizes			
Parameters	256	512	1024	2048
K	0.004001	0.004527	0.005746	0.006279
r	2.689487	2.112879	1.498743	2.505025
t_0	2.354282	2.336269	2.540356	2.985071

Table 2: Logistic function parameters for each packet size

It was observed that the parameter K and t_o in these closed form expressions linearly increase when the packet size exponentially rises as a function of the power two. The transformed packet size results in a linear function as

$$\text{linear value} = \log_2^{\text{packetsize}} \tag{4.2}$$

Evaluating values produces the results shown in Table 3

	Linear value transformation of packet sizes			
Parameters	8	9	10	11
K	0.004001	0.004527	0.005746	0.006279
r	2.689487	2.112879	1.498743	2.505025
t_0	2.354282	2.336269	2.540356	2.985071

Table 3: Logistic function parameters for each linear value of packet size

It is observed that parameter K and t_o could be presented as a polynomial function of linear value of packet sizes. The basic fit tool box in the Matlab simulator [48] was used to develop a best fit model for the values. Figure 29 shows the three different models of K, i.e. linear function, quadratic function, and cubic function, plus their residuals. Even though the cubic function gave smaller residuals, the closed from expressions given were negative functions. The quadratic function also gave too high rate of change for K.

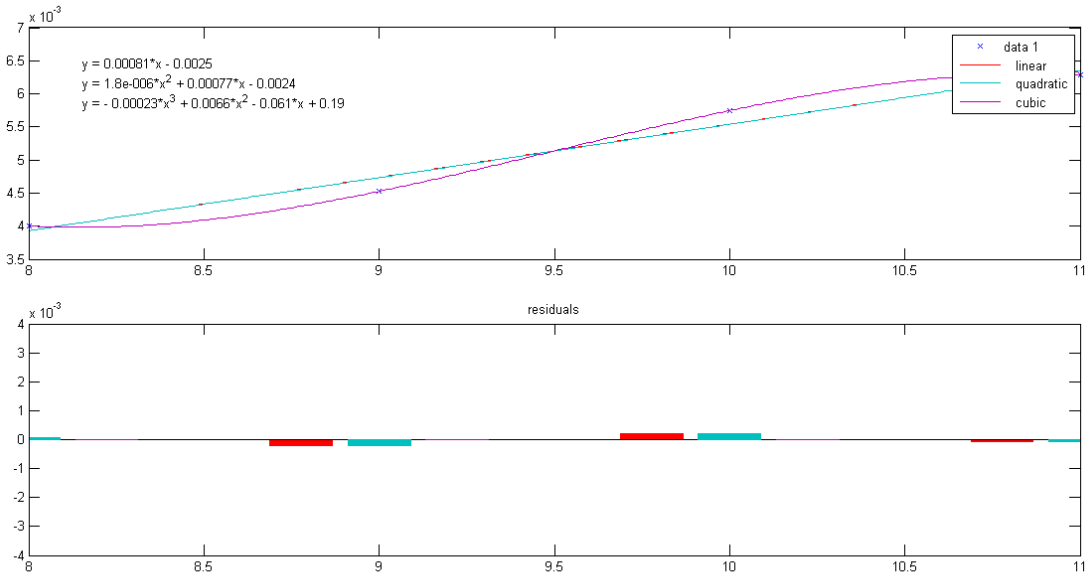

Figure 29: Graph of K vs. Linear value of packet size

A linear function gave the best fit K values, the result of which was

$$y = 0.00081x - 0.0025 \quad (5.3)$$

which presents K as a function of packet size as

$$K = 0.00081\left(\log_2^{packetsize}\right) - 0.0025 \tag{5.4}$$

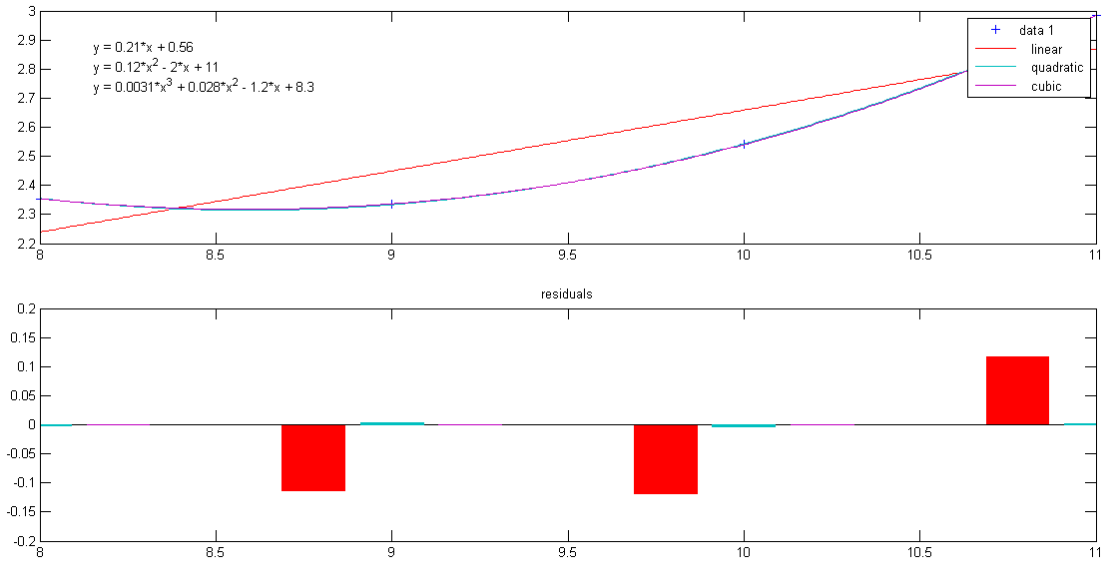

Figure 30: Graph of t_o vs. Linear value of packet size

A similar method was performed with the t_0 parameter. The result is shown in Figure 30. A linear function gave the best fit for t_0 values as

$$y = 0.21x + 0.56 \tag{4.3}$$

which forms a function of t_0 resulting from packet size as

$$t_0 = 0.21\left(\log_2^{packetsize}\right) + 0.56 \tag{4.4}$$

The value *r* could not be represented by any polynomial. The best possible parameter for solution in this case was a mean value of them, which gave a value equal to 2.2015. With e.q. 4.1 to e.q 4.4, the closed form expression which best approximated the delay in terms of number of components and packet size was given by the following expression.

$$D(\text{number of component, packet size}) = \frac{K}{1+e^{-r(\text{number of component}-t_0)}} \quad (4.5)$$

where

$$K = 0.00081\left(\log_2^{\text{packetsize}}\right) - 0.0025$$
$$r = 2.2015$$
$$t_0 = 0.21\left(\log_2^{\text{packetsize}}\right) + 0.56$$

K : the upper bound of the delay.
r : the growth rate of delay, in exponential region.
t_0 : where the maximum growth appear.

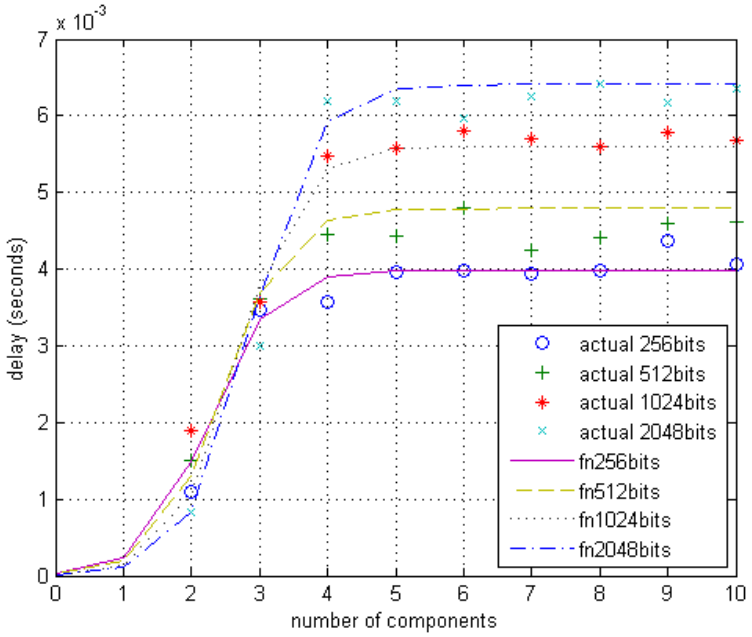

Figure 31: Graph of average total link delay vs. Number of components of 4 different packet sizes

The closed form expression in e.q. 4.5 was plotted with the actual data in Figure 31. In Figure 32, the absolute errors between actual delays and closed form expression were plotted. Figure 33 was a plot of the relative error of them in percentage. Even though this model addressed a very high error when number of components is small, the maximum error was as high as 44%, the model performed very well when the number of components is large, 0-10%.

The closed form expression of the delay, in terms of number of components and packet size, can be used for designing a better SCA-SDR waveform. It can be used in predicting the link delay related from number of components and packet size in the SCA-SDR system. The model also gives more realistic delay in simulation which leads to more practical results.

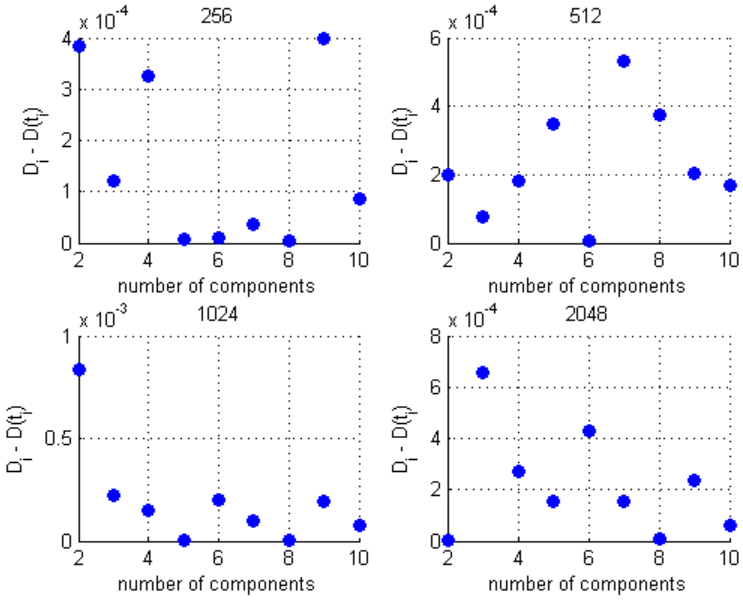

Figure 32: Graph of Absolute error vs. Number of components of four different packet sizes

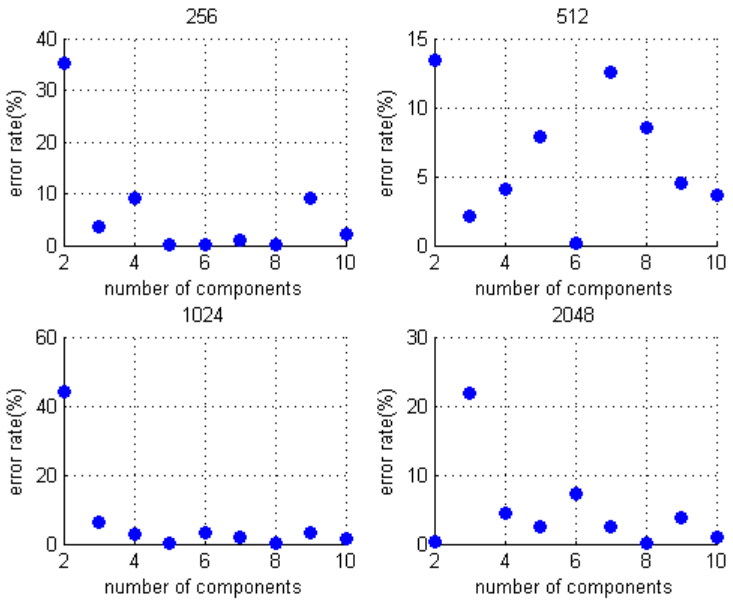

Figure 33: Graph of relative error in percentage (%) vs. Number of components of four different packet sizes

The experiment was re-run with additional packet size, 4096 bits, to test the validity of this model. The closed form expression in e.q. 4.5 was plotted with the actual data in Figure 34. In Figure 35, the absolute errors between actual delays and the closed form expression were plotted. Figure 36 showed the relative error between them. The model still performed very well even though the number of components was increased. Based from these figures, the model still holds true even when the packet was raised to 4096 bits.

The closed form expression can also be used with other CPU and memory consumption models to choose the number of components and the packet size in the waveform to satisfy protocol limitation and to optimize the throughput.

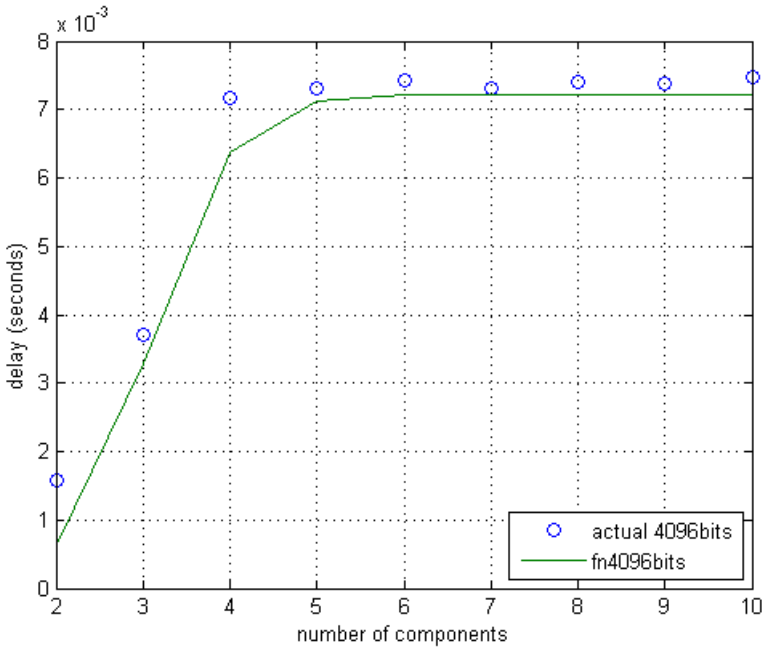

Figure 34: Graph of average total link delay vs. Number of components of 4096 bits packet sizes

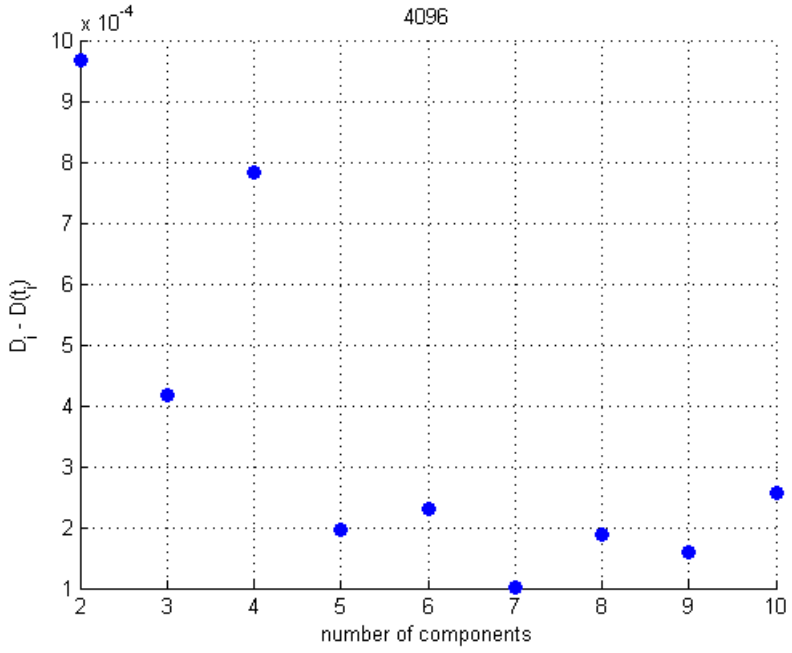

Figure 35: Graph of Absolute error vs. Number of components of 4096 bits packet size

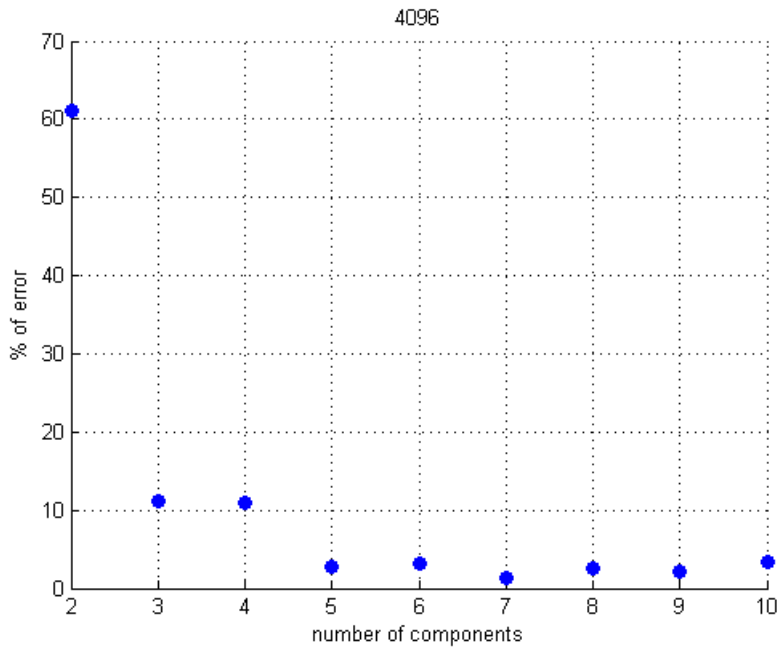

Figure 36: Graph of relative error in percentage (%) vs. Number of components of 4096 bits packet size

4.5. Conclusion

In this chapter, a study of the internal inter-component communication delay due to number of components and packet size was presented. Through controlled environment investigation, it was demonstrated that the increase of total link delay was not linear with packet size. A closed form expression is proposed using the logistic function to model the behavior of the delay as a function of number of components and packet size. The accuracy of this model is presented by comparing it with actual data. . The model had less accuracy when the number of components was small, however it was better when the number of components was large.

CHAPTER 5

ENERGY AWARE WAVEFORM SELECTION ALGORITHM FOR SDR

The goal of this work is to develop a new approach for a SDR transmission protocol that optimizes energy efficiency. In order to choose among multiple protocols, a requirement metric for algorithm selection was formed. The elements of this metric are availability, quality of service, security, and user personal requirement. To design the next generation SDR, one parameter to be included in this metric is energy consumption. To achieve that result, the level of energy consumption of each protocol must be predicted in real-time in order to identify the best possible protocol solution.

This chapter is organized as follows: In section 5.1, an energy model and selection algorithm is discussed. In section 5.2, simulation setup, environment assumption, and performance index are discussed and given. Results and analysis with mathematical modeling are presented and discussed in Section 5.3. In section 5.4, the conclusion is given.

5.1. Energy Model and Selection Algorithm

Three energy models were derived by Kim et al. [49]. These models represent energy consumption of three different whole traffic wireless network protocols – WLAN [50], Bluetooth [51], and ZigBee [52].

Wireless Local Area Network (WLAN) or WLAN is one of the most well-known and widely used wireless technologies in the past decades. WLAN is the technology based on traditional LAN but instead of using wire, the architecture shifted to wireless.

WLAN itself has many different standards i.e. 802.11b and 802.11g use the 2.4 GHz band, 802.11a uses 5 GHz band, 802.11n uses both 2.4 GHz and 5 GHz band depends on the speed and range, and 802.11ac uses the 5 GHz band. The transmission rate of WLAN also depends on the standards, the range is 11 Mbit/s up to 6.93 Gbit/s, in idle. WLAN uses an ad-hoc model so any device can transmit at any time. Due to this reason whenever the WLAN device needs to transmit a data, the channel has to be free for a long enough period before the transmission starts. If a collision occurs, all the devices involved in the collision will wait for a random time before retransmitting. This process is called “back-off". The energy model of WLAN [49] is, $L_{traffic}$ which is a length of total transmitted traffic.

$$E_{WLAN}=\left[P_{tx}^{WLAN}\cdot\frac{L_{MTU}^{WLAN}}{R^{WLAN}}+P_{rx}^{WLAN}\cdot\frac{L_{ACK}^{WLAN}}{R^{WLAN}}+P_{idle}^{WLAN}\cdot(T_{DIFS}^{WLAN}+T_{BO}^{WLAN}+T_{SIFS}^{WLAN})\right]\cdot\frac{L_{traffic}}{L_{payload}^{WLAN}} \quad (5.1)$$

with

E_{WLAN}= energy consumption of WLAN

P_{tx}^{WLAN}= transmission power of WLAN

P_{rx}^{WLAN}= receiving power of WLAN

P_{idle}^{WLAN}= power consumption of WLAN in idle state

L_{MTU}^{WLAN}= Maximum Transmission unit of WLAN

L_{ACK}^{WLAN}= Length of WLAN acknowledgement

$L_{payload}^{WLAN}$= Length of actual payload per MTU

R^{WLAN}= Rate of transmission of WLAN

T_{DIFS}^{WLAN}= The channel-free guarantee time, the minimum time that channel has to be continuously free before start transmission to prevent collision. WLAN uses Distributed coordination function (DCF) Interframe space

(DIFS) as the channel-free guarantee, which is Shot Interframe Space (SIFS) plus two time slots.

T_{BO}^{WLAN}= The additional waiting time adds in WLAN when it detects the collision.

T_{SIFS}^{WLAN}= The maximum period that WLAN would wait for the acknowledgment, which comes from Shot Interframe Space (SIFS).

Bluetooth is a wireless network standard based on IEEE 802.15.1 (Wireless Personal Area Network – WPAN). The main purpose of this standard is used for exchanging data over short distances, 1-100 meters. The transmission rate of Bluetooth is around 1 – 24 Mbits/s, depending on the version. Bluetooth also uses the 2.4 GHz band but it is set up as a master-slave model. So the devices in same network don't need to sense or be aware of collisions. All devices will be given a specific transmit time-slot from the Master device to transmit. The energy model of Bluetooth [49] is

$$E_{BT}=\left[P_{tx}^{BT}\cdot\frac{L_{MTU}^{BT}}{R^{BT}}+P_{rx}^{BT}\cdot\frac{L_{ACK}^{BT}}{R^{BT}}+P_{idle}^{BT}\cdot\left(T_{slot}^{BT}-\left(\frac{L_{MTU}^{BT}}{R^{BT}}+\frac{L_{ACK}^{BT}}{R_{ACK}^{BT}}\right)\right)\right]\cdot\frac{L_{traffic}}{L_{payload}^{BT}} \quad (5.2)$$

with

E_{BT}= energy consumption of Bluetooth

P_{tx}^{BT}= transmission power of Bluetooth

P_{rx}^{BT}= receiving power of Bluetooth

P_{idle}^{BT}= power consumption of Bluetooth in idle state

L_{MTU}^{BT}= Maximum Transmission unit of Bluetooth

L_{ACK}^{BT}= Length of Bluetooth acknowledgement

$L_{payload}^{BT}$= Length of actual payload per MTU of Bluetooth

R^{BT}= Rate of transmission of Bluetooth

T_{slot}^{BT}= Time of Bluetooth time slot, 625 microseconds.

Zigbee is another wireless network standard based on IEEE 802.15.4 for Low-Rate Wireless Personal Area Networks (LR-WPANs). The purpose of this standard is to be simpler, less expensive, and lower-power. Zigbee has been designated as a low-rate application device such as in-home sensor or wireless home appliance. Zigbee therefore transmits with 250 kbit/s which is a lower rate than the other protocols listed above. Zigbee can also operate in an unlicensed 2.4 Ghz band, along with other unlicensed bands depending on the country it is used in. The significance of ZigBee is that it can operate as mesh network so each ZigBee device can act as a router that route the packet to other devices. ZigBee also uses an ad-hoc model so any device can transmit at any time but using different methods to solve the collision problem than WLAN. Instead of using a statistical channel free model, a ZigBee device will allocate the free channel by broadcasting the allocating signal to all others. The energy model of ZigBee [49] is

$$E_{ZB}=\left[P_{tx}^{ZB}\cdot\frac{L_{MTU}^{ZB}}{R^{ZB}}+P_{rx}^{ZB}\cdot\frac{L_{ACK}^{ZB}}{R^{ZB}}+P_{idle}^{ZB}\cdot(T_{BO}^{ZB}+T_{CCA}^{ZB}+T_{ACK}^{ZB})\right]\cdot\frac{L_{traffic}}{L_{payload}^{ZB}} \quad (5.3)$$

with

E_{ZB}= energy consumption of ZigBee

P_{tx}^{ZB}= transmission power of ZigBee

P_{rx}^{ZB}= receiving power of ZigBee

P_{idle}^{ZB}= power consumption of ZigBee in idle state

L_{MTU}^{ZB}= Maximum Transmission unit of ZigBee

L_{ACK}^{ZB}= Length of ZigBee acknowledgement

$L_{payload}^{ZB}$= Length of actual payload per MTU of ZigBee

R^{ZB}= Rate of transmission of ZigBee

T_{BO}^{ZB}= The additional waiting time adds in ZigBee when it detects the collision.

T_{CCA}^{ZB}= The clear channel assessment (CCA) time, the minimum time that channel has to be continuously free before start transmission to prevent collision.

The additional waiting time adds to the protocol when WLAN detects the collision.

T_{ACK}^{ZB}= The maximum period that WLAN would wait for the acknowledgment.

Kim et al. and Seigneur. J et al. [53] obtained the states of power consumption for each protocol. In Table 1, the power consumption of the three protocols are shown normalized that of to the WLAN inactive state.

	WLAN	**Bluetooth**	**ZigBee**
Inactive	1x	0.30x	0.03x
Idle	2.14x	0.32x	0.21x
Tx	3x	0.57x	0.21x
Rx	2.71x	0.57x	0.21x

Table 4: Normalized Power Consumption between WLAN, BLUETOOTH, and ZIGBEE

As shown in Figure 37, the power aware module must work side-by-side with the SCA core framework. By working together, the system should make a decision in advance in the terms of power consumption, i.e. predetermine which waveform saves more power when many waveforms are available. The parameters which have to be considered are required rate, actual rate, data availability, maximum transmission unit (MTU) of protocol, header length, and swapping cost.

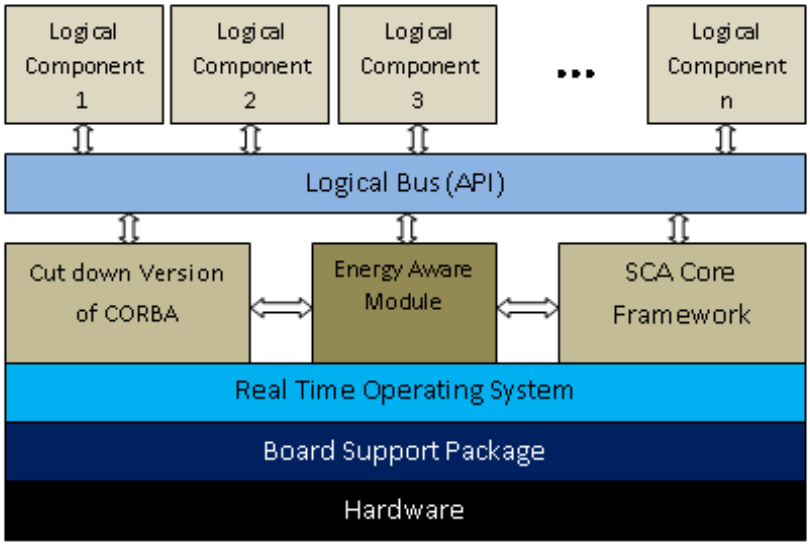

Figure 37: Energy Aware Module for SCA

Figure 38 shows the algorithm flowchart for this module. When the application requests to transmit, it will check whether a specific waveform is required for its transmission or not. If the application requires a specific choice, it will select the waveform immediately. Otherwise, it will compare the required data rate of the application with every waveform. Any waveform which can support the application will be added to the candidate list, $W_{candidate}$. This specific step has been performed to check the quality of service needs for each application. Then, the application will be checked again if it is a real-time application or not. If it is, the candidate with the lowest rate will be used to transmit. This is because the real-time application has a different characteristic compared to the normal application. Real-time data have to be received in sequence and the data will not be available ahead of time. If the application does not need real-time

capability, energy consumption of each of them will be calculated using e.q. 5.1 - 5.3. The energy cost that occurs when changing waveforms is also taken into consideration here. The waveform with the lowest energy consumption will be chosen to transmit, and the process will continue through the next application request.

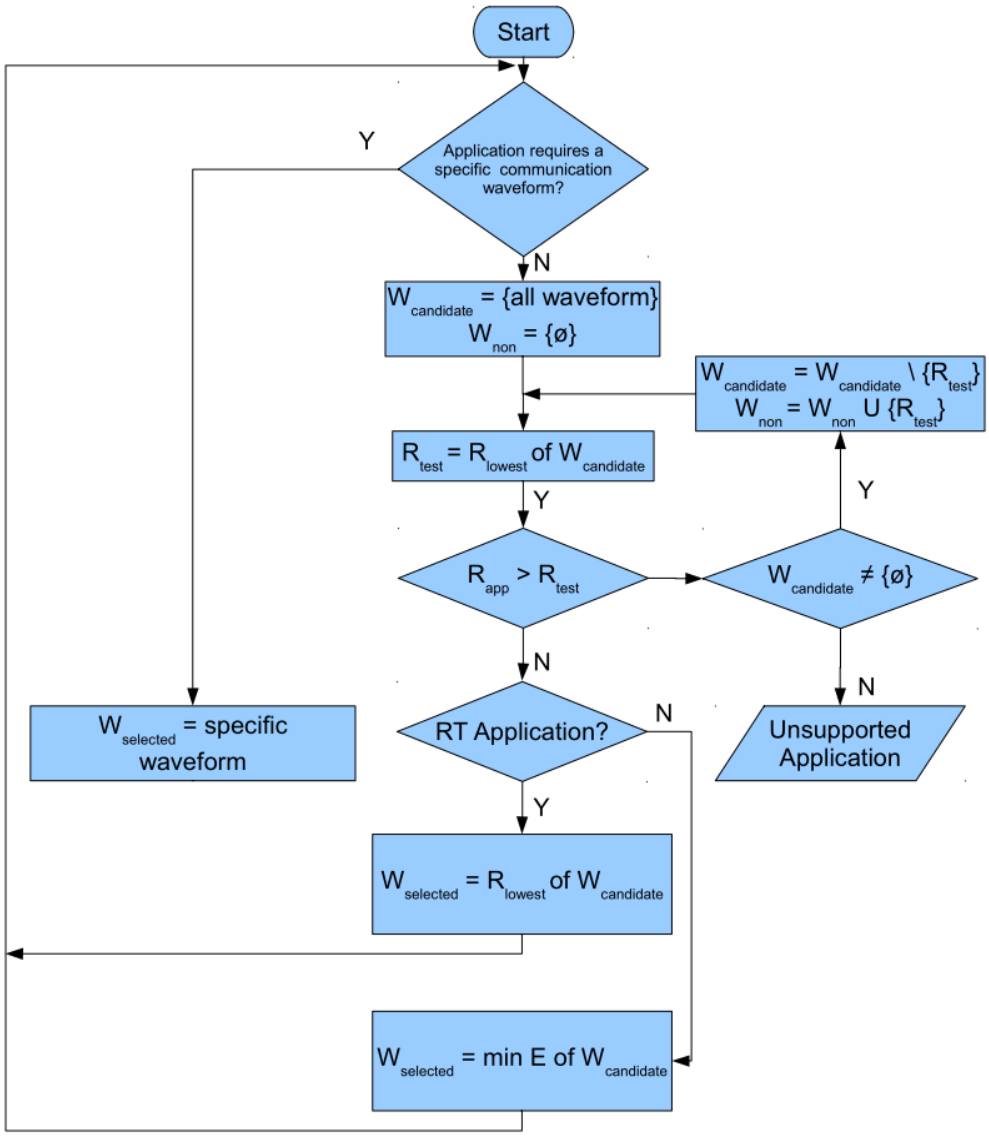

Figure 38: Energy Aware Waveform Selection Scheme

This algorithm was evaluated through simulation. The simulation was performed in one-hundred steps. In each step, one thousand uniformly distributed random application traffic length values were used. The average value for the first step was five hundreds bytes. Afterwards, the average traffic length value for each step was increased by one thousand bytes.

5.2. Simulation Setup, Environment, and Performance Index

The simulation environment was the direct peer-to-peer communication between two network entities. The distance between the two entities was within the range of all waveforms. Each single network entity was equipped with three different waveforms – WLAN, Bluetooth, and ZigBee. Also, the channel was assumed to be only available to these two entities. WLAN was specific to IEEE 802.11g standard. Bluetooth was specific to Data High rate (DH) 1-28 bytes per time slot, symmetric link-single time slot. ZigBee was a Non-Beacon enable mode with symmetric uplink and downlink. The performance index of this scheme is energy consumption. The energy consumption of the proposed system was compared to that of each standalone communication device.

5.3. Results and Analysis

As shown in Figure 39, the energy consumption of each protocol is proportional to the traffic length. Obviously, more energy is required for more data transmission. The graph also shows that energy increases as a step function.

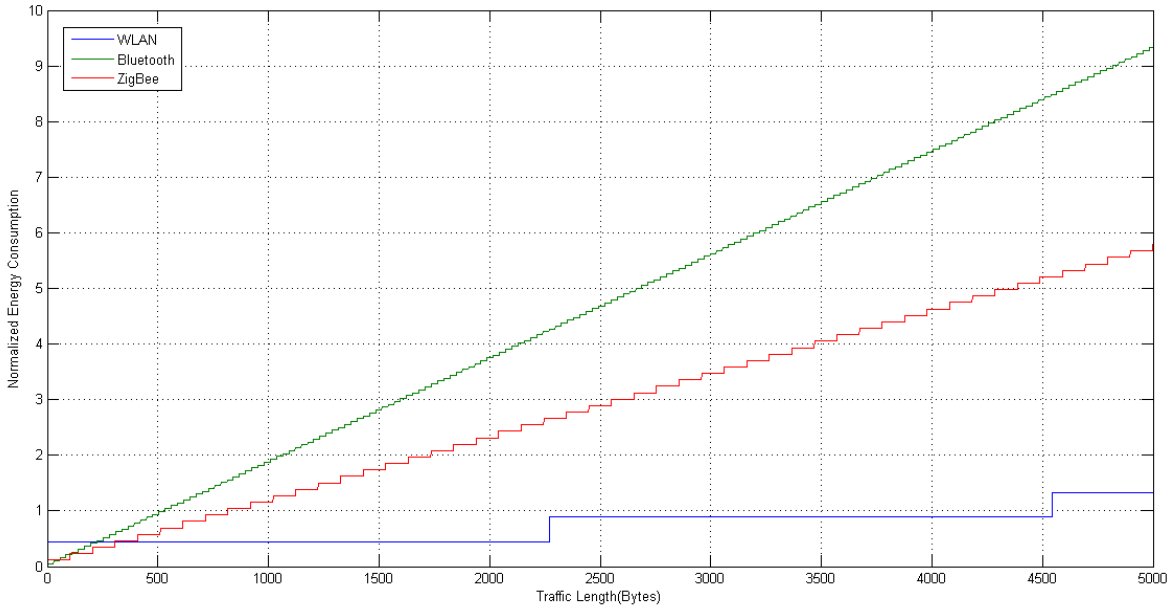

Figure 39: Energy Consumption vs. Traffic Length

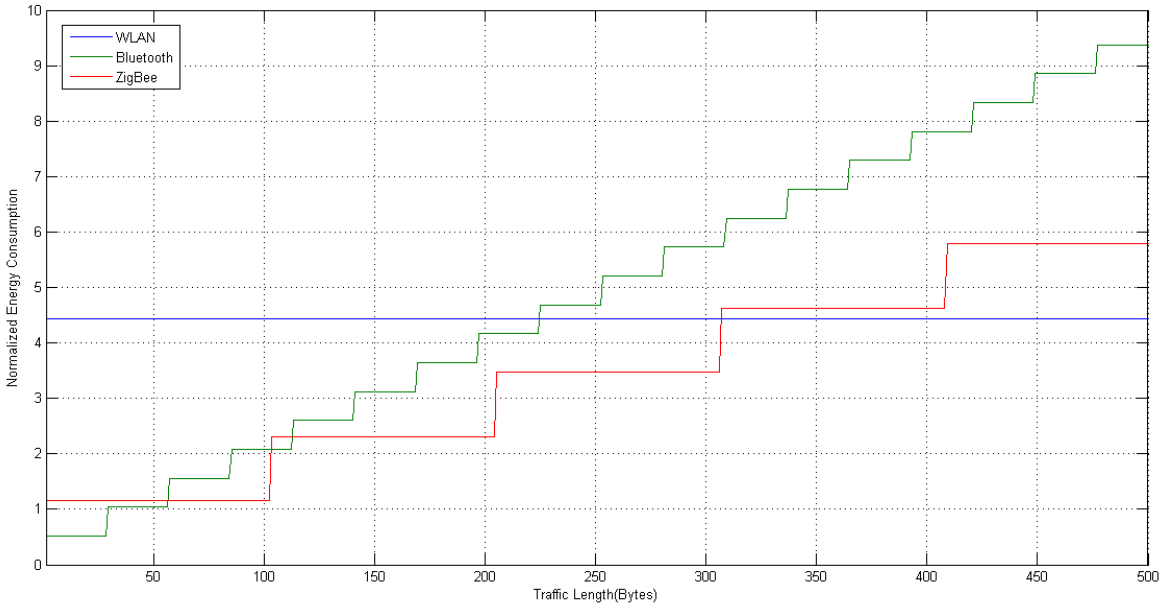

Figure 40: Energy Consumption vs. Traffic Length zoomed on the first five hundred bytes.

The step functions occur because each packet transmission has to transmit the whole packet regardless of the completeness of the packet. This leads to a better selection of application data size which is based on energy consumption. The application needs to limit or compress transmissions data close to the packet size to fully utilize the energy consumption per packet. Besides, this comparison study shows one significant concern. Even though the low power protocol devices, like Bluetooth and ZigBee, consume less power to operate, energy consumption due to traffic length grows much faster than that of WLAN. As show in Figure 40, for the first 500 bytes, Bluetooth is the most energy efficient when the traffic length is between 1 – 200 bytes. If the traffic length is between 200 – 300 bytes, ZigBee becomes the most energy efficient.

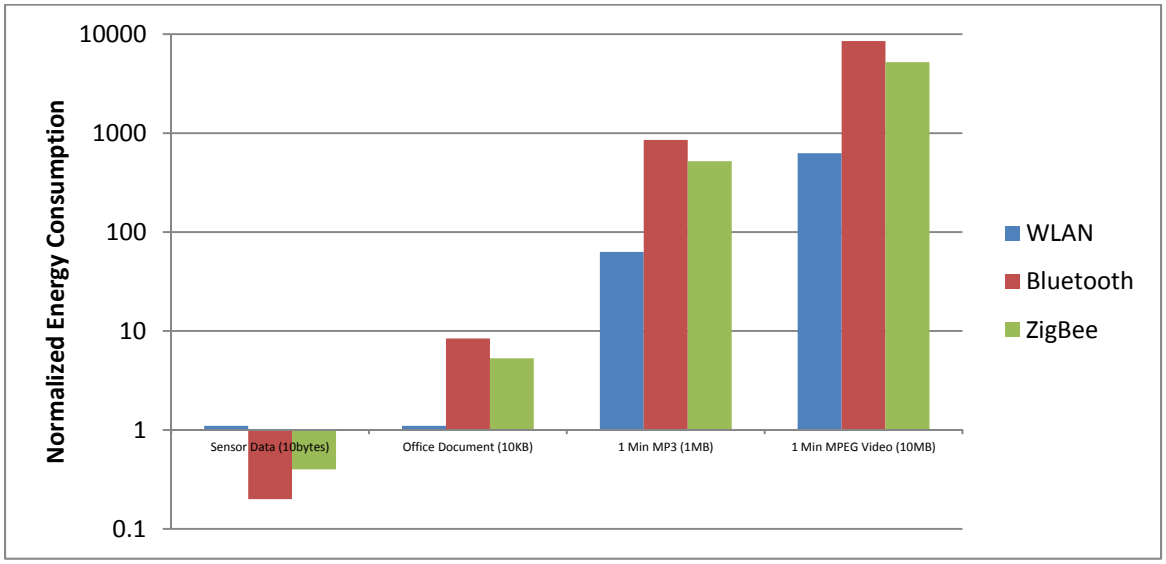

Figure 41: Example of Real Life Application Traffic length: Energy Consumption vs. Traffic Length

Above that range, WLAN performs the best. The reason behind this is that the MTU of each protocol is different. Bluetooth has the smallest MTU size compared to

ZigBee and WLAN. When more packets are transmitted, more energy is wasted on overhead. To utilize energy more efficiently, we must select waveforms based on the MTU size, header size, transmission power, and transmission rate as well as the availability of data. For a real-time application which does not have the whole information on the traffic length, the required rate can be considered as traffic length instead. Figure 41 shows the energy consumption of each protocol in terms of real-life application traffic lengths. It can be concluded that if the application needs to transmit a huge file, like audio or video, WLAN is more desirable.

5.3.1. Performance of the proposed Scheme

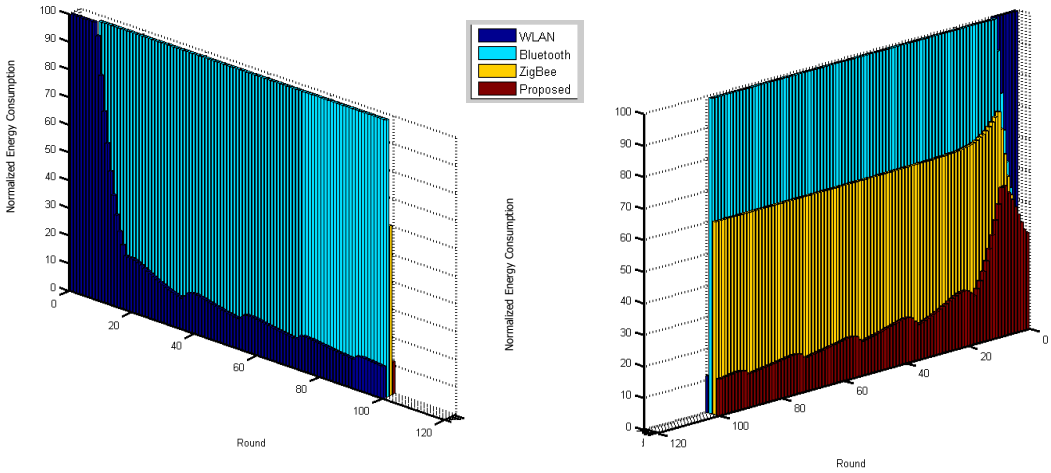

Figure 42: Normalized Energy Consumption of short traffic length Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Proposed Scheme

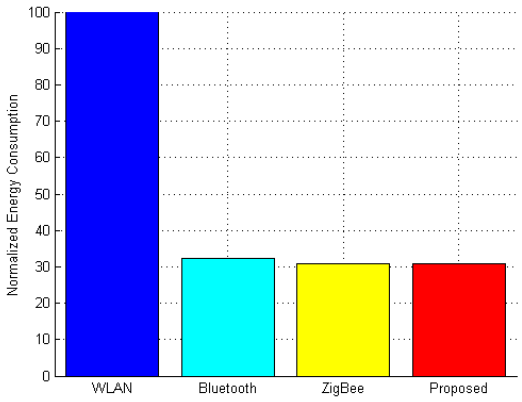

Figure 43: Normalized Energy Consumption of short traffic length. Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Proposed Selection Scheme

To evaluate the performance, random traffic lengths were used as mentioned previous. The proposed scheme performance was evaluated and compared to each traditional protocol. The graph figures in this section use color codes to represent the different waveforms: dark blue for WLAN, light blue for Bluetooth, yellow for ZigBee, and Red for the proposed scheme.

Figure 42 shows a trend that the proposed algorithm will transmit by using a waveform that causes minimum energy. The average energy savings were 30%, 81%, and 50% compared to WLAN, Bluetooth, ZigBee, respectively. Figure 43 shows the energy consumption for the first one thousand bytes. For WLAN, the energy consumption is much higher than the other waveforms because the average transmission length is short. A single traffic can be transmitted by only a single packet of Bluetooth or ZigBee because ZigBee has the smallest header and the packet size matches transmission lengths,

therefore it consumes least energy. On the other hand, WLAN will perform better if the traffic length is long.

5.3.2. Limitation of the proposed Scheme

Although the proposed scheme provides a good solution in selecting a waveform to transmit data in terms of energy efficiency, an assumption that input stream has uniform random distribution lengths was not realistic. In real world application, mobile entities do not have a uniform random distribution of traffic length. Every entity has its own specific purpose and often transmits application data with similar length. For example in mobile sensor networks [54-56], the sink frequently transmits and receives short-length sensor data but rarely uploads data to the backup server. The shortcoming of the proposed scheme was observed using a non-uniform random distribution of traffic lengths. Two extreme cases were shown. In the first case, the distribution was skewed towards a long traffic length while in the second case short traffic lengths were given more weight.

The result of the first case is shown in Figure 44. The total energy consumption of the proposed scheme is much higher when compared with the traditional WLAN. When the short traffic length occurs, and the waveform is switched to the low-power protocol in order to match the MTU size, the cost of switching back to WLAN protocol is excessively high and the real-time decision does not take into account the future traffic lengths. In this situation, the used of WLAN would result in better energy efficiency in the long term.

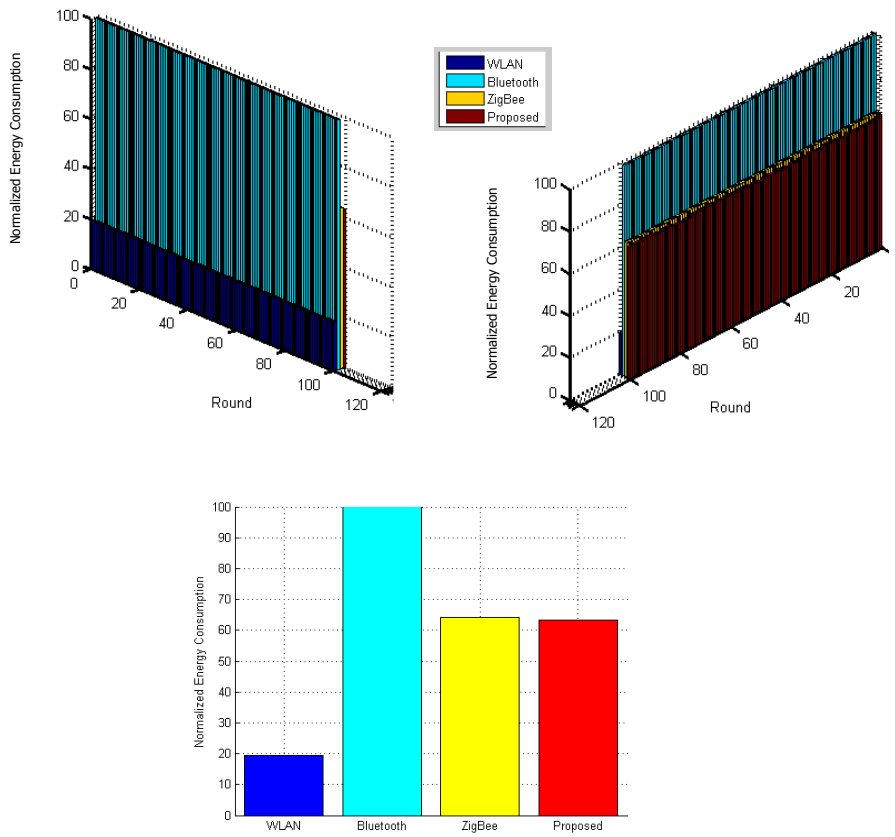

Figure 44: Normalized Energy Consumption with high probability (> 0.75) of long traffic length (> 2000 bytes). Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme

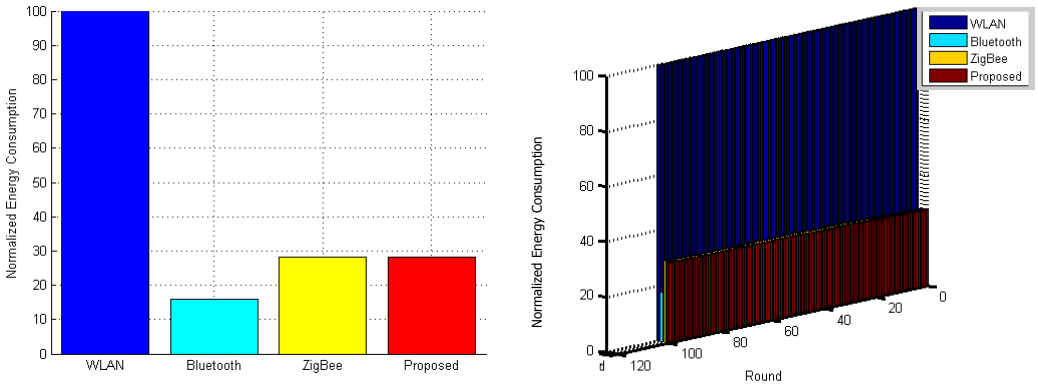

Figure 45: Normalized Energy Consumption with high probability (> 0.75) of short traffic length (< 200 bytes). Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme

Figure 45 also shows the other extreme case. The balance was flipped to the short traffic length and, as a result, the proposed scheme consumed more energy than Bluetooth. The reason behind it is also similar to the previous case. The cost of switching the protocol is too high, therefore, this resulted in the system continuing to transmit with WLAN. Recognizing these two extreme cases could lead to a better algorithm design.

5.3.3. Threshold value

To overcome a limitation in the scheme, the additional procedures were added to the selection algorithm. As shown in Figure 46, a counter had been added to a scheme. Instead of immediately changing protocol, the counter kept counting every time a decision was made to change the protocol, until it reached the threshold, so it would change the protocol and reset the counter.

A question is what should be the value of this threshold should be. A simulation was performed again with an extended version of the scheme over the same performance index. The threshold value was varied from one to a maximum set of information transmitted minus one (n-1). The simulation was performed with two additional extreme cases as discussed before. Figure 47 shows the result of the first simulation where the majority of traffic sizes were large. Figure 48 showed the result of the second simulation where the majority of traffic sizes was small. These results show a conclusion that if there was no threshold at all, in extreme cases the selection algorithm performed worse than without. The scheme would starts to perform better after setting the threshold value equal two in those two extreme cases.

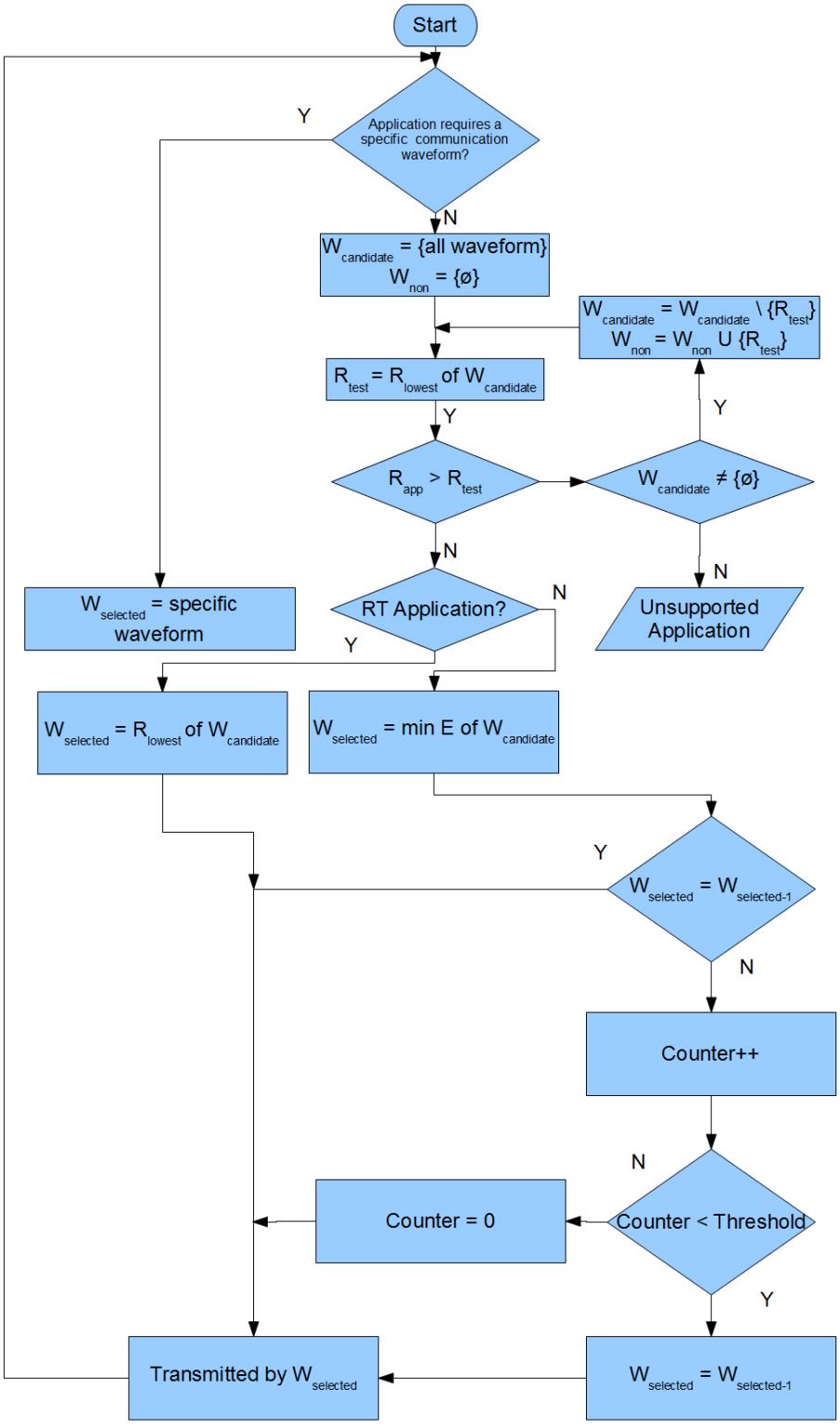

Figure 46: Smart Energy Aware Waveform Selection Scheme

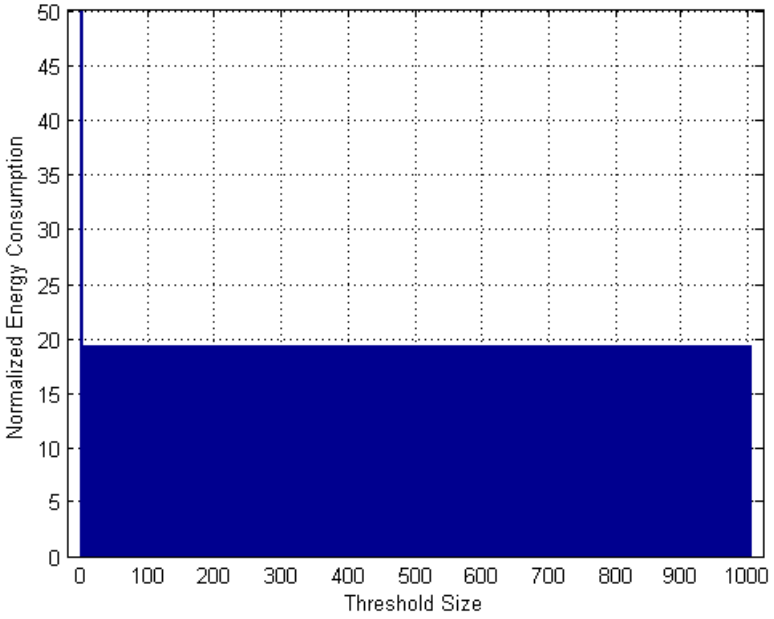

Figure 47: Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of long traffic length (> 2000 bytes).

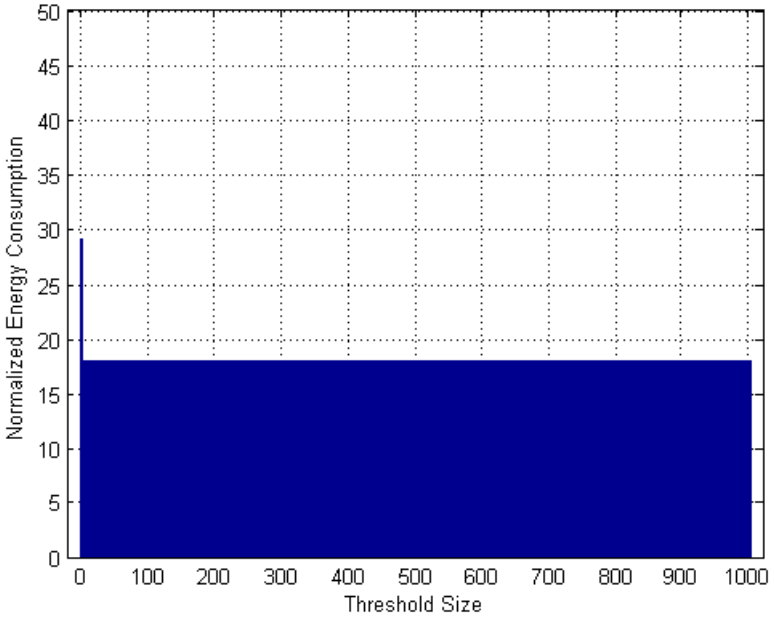

Figure 48: Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of short traffic length (< 200 bytes).

Figure 49 showed the performance in terms of the energy consumption of the scheme using threshold value in three different scenarios, compared with three traditional transmissions. The first scheme, a normal distribution of traffic sizes, was presented with the blue bar graph. The second and third schemes, represented by green and red bar graphs, were comprised of the case where the majority of traffic sizes were small and large, respectively. Even though the scheme did not give the lowest energy consumption in all cases, the performance was 10-15% better than without a threshold.

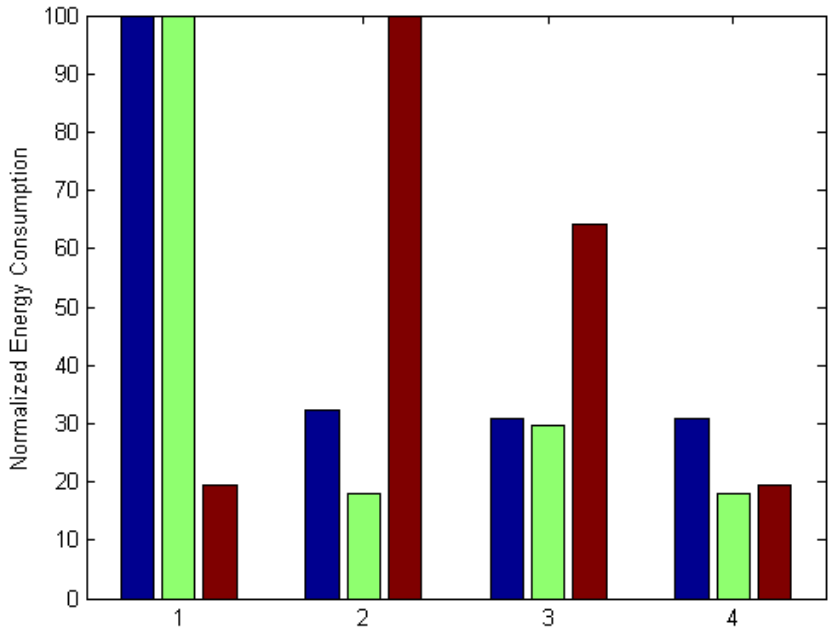

Figure 49: Normalized Energy Consumption of "normal distribution traffic length (blue)" ,"high probability (> 0.75) of short traffic length (< 200 bytes) (green)", and "Graph of Normalized energy consumption vs. Size of threshold value for high probability (> 0.75) of long traffic length (> 2000 bytes) (red)". Comparison between (1) WLAN Only (2) Bluetooth Only (3) ZigBee only (4) Energy Aware Selection Scheme with threshold changing

5.4. Conclusion

A comparison study of energy consumption between three wireless network protocols is presented. This comparison illustrates that even though a low-power protocol device was used, it did not necessarily consume less energy transmitting the same traffic length. This is because energy was consumed transmitting unnecessary duplicate headers. An energy aware waveform selection algorithm for SDR is proposed here. Evaluation of the proposed scheme was conducted through simulations. Investigation results show that the proposed scheme performed better than using any single protocol specific to an application. Limitation of the scheme was also addressed when used with non-uniform random distributions of traffic lengths. An additional procedure was added to the scheme to improve the limitation.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

In this dissertation, the relationship between total link delay versus the number of components and packet sizes in a controlled environment were demonstrated. The results show that the delay will not linearly increase with increasing numbers of components. Instead, the delay will saturate after a certain number of components in the waveform is reached. This response occurs because the delay is overcome by the communication setup, especially related to the CORBA itself. The delay also does not proportionally increase with the packet size. The results demonstrate that the delay will logarithmically increase due to packet size. This response results because the majority of the delay comes from the CORBA processing overhead which does not increase due to packet size. The relationship between packet size and number of components was also observed. Packet size has less effect on the delay when the number of components is small. A closed form expression was selected to model the total link delay in terms of the number of components and the packet size as e.q. 4.5. The accuracy of the expression was investigated. The model performed well when the number of components was large but not as well when the number of components was small. The closed form expression is useful to predict delay in terms of those two parameters which creates a better design methodology SDR-SCA based waveform.

Possible future work can investigate an extended experiment using advanced controlled environments. This will allow evaluating another effect due to the number of components, especially with regards to multi-core or special designed core systems. Also

the same investigation methods can be used to test goodness of fit of the designed platform in term of the number of components and the packet size. This investigative approach can also be extended to other resource consumption, CPU, memory, power models. This will help to determine a better model relating the number of components and waveform packet size versus protocol limitation and throughput optimization.

Lastly, the energy waveform selection algorithm was proposed in order to optimize the energy consumption for SDR transmission. A flexible SDR system can support multiple transmission protocols at the same time. It needed to be seamless from the user to choose which protocol to be transmitted. The goal of the scheme was to guarantee the availability of the service and to keep the battery lasting longer. The scheme used the three widely used protocol energy models [49] to predict the energy consumption of the system plus the switching consumption. The simulation compared the goodness of the scheme over those three traditional non switching transmissions. The scheme limitation was also addressed when the probability of traffic length was distributed unfairly. The additional switching threshold was also proposed to increase the performance of the scheme.

A possible future research direction is to extend the energy analysis model to other wireless protocols. Currently the scheme only focused on WLAN, Bluetooth, and ZigBee but the concept itself was general enough to be extended to other waveforms such as GPRS, EDGE. Moreover, this scheme can be more adaptive. The energy selection algorithm could be improved by adding the adaptive prediction concept. Using "Knowledge" based ideas, even perfect knowledge or heuristic knowledge, the scheme

could perform better in extreme cases. The threshold can be dynamically changed depending on the ratio of traffic size that needs to be transmitted. Also the additional statistical model could be considered on user's behavior for application specific equipment. This can lead to a better decision metric for waveform selection scheme.

REFERENCES

[1] J. Mitola, “Cognitive Radio Architecture Evolution,” *Proceedings of the IEEE*, vol. 97, no. 4, pp. 626–641, Apr. 2009.

[2] IEEE 802.22-2011 Specification, Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Policies and procedures for operation in the TV Bands, 01 July 2011.

[3] R. W. Thomas, L. A. DaSilva, and A. B. MacKenzie, “Cognitive networks,” in 2005 First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005, 2005, pp. 352–360.

[4] “Software Communications Architecture Specification,” Joint Tactical Radio System (JTRS) Joint Program Office, Version 2.2.2, May 2006.

[5] “Common Object Request Broker Architecture (CORBA) Specification,” Object Management Group, Version 3.1, January 2008.

[6] G. Ismael, mez, M. Vuk, S. Jose, and G. Antoni, "A Lightweight Operating Environment for Next Generation Cognitive Radios," *in Proceedings of the 2008 11th EUROMICRO Conference on Digital System Design Architectures*, Methods and Tools: IEEE Computer Society, 2008.

[7] C. A. Linn, "A simple, lightweight communications architecture facilitating SCA application portability," *in SDR Technical Conference and Product Exposition*, Washington, DC, 2009.

[8] I. Gomez, V. Marojevic, J. Bracke, and A. Gelonch, “Performance and overhead analysis of the ALOE middleware for SDR,” *in MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, 2010, pp. 1134–1139.

[9] I. Gomez, V. Marojevic, and A. Gelonch, “Aloe: An open-source SDR execution environment with cognitive computing resource management capabilities,” *IEEE Communications Magazine*, vol. 49, no. 9, pp. 76–83, Sep. 2011.

[10] Richardson, K. Jimenez, C. Stephens, and D.R., "Evolution of the software communication architecture standard," *in Military Communications Conference MILCOM 2009*. IEEE Boston, MA, 2009.

[11] G. Jianxin, Y. Xiaohui, G. Jun, and L. Quan, "The Software Communication Architecture specification: Evolution and trends," *in Computational Intelligence and Industrial Applications, 2009*. *PACIIA 2009* Wuhan, China, 2009.

[12] T. Ulversoy, "Software Defined Radio: Challenges and Opportunities," *in IEEE Communications Surveys & Tutorials*. vol. 12, 2010, pp. 531 - 550.

[13] J. Tang, S. Liu, Z. Gu, C. Liu, and J.-L. Gaudiot, “Prefetching in Embedded Mobile Systems Can Be Energy-Efficient,” *Computer Architecture Letters*, vol. 10, no. 1, pp. 8–11, Jun. 2011.

[14] M. R. Philip J. Balister, Jeffrey H. Reed, "Impact of the use of CORBA for Inter-Component Communication in SCA Based Radio," *in SDR Technical Conference and Product Exposition*, Orlando, FL, 2006.

[15] T. Tsou, P. Balister, and J. Reed, "Latency Profiling for SCA Software Radio," *in SDR Technical Conference and Product Exposition,* Denver, CO, 2007.

[16] P. J. Balister, C. Dietrich, and J. H. Reed, "Memory Usage of a Software Communication Architecture Waveform," *in SDR Technical Conference and Product Exposition*, Denver, CO, 2007.

[17] G. Abgrall, F. L. Roy, J.-P. Delahaye, J.-P. Diguet, and G. Gogniat, "A Comparative Study of Two Software Defined Radio Platforms," *in SDR Technical Conference and Product Exposition*, Washington, D.C., October 26-30 2008.

[18] Gael Abgrall, F. e. e. L. Roy, J.-P. Diguet, G. Gogniat, and J.-P. Delahaye, "Predictibility of Inter-component latency *in a Software Communications Architecture Operating Environment," in IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW),* Atlanta, GA, 19-23 April 2010.

[19] J. H. Reed, Software Radio: A Modern Approach to Radio Engineering, 1st ed. Prentice Hall, 2002.

[20] P. Johnson, “New Research Lab Leads to Unique Radio Receiver,” E-Systems Team, vol. 5, no. 4, pp. 6–7, May 1985.

[21] Peter Hoeher and Helmuth Lang, “Coded-8PSK modem for fixed and mobile satellite services based on DSP,” *in First International Workshop on Digital Signal Processing Techniques Applied to Space Communications*, ESA/ESTEC, Noordwijk, Netherlands, 1988, vol. January 1990, pp. 117–123.

[22] J. Mitola, III, “Software radios-survey, critical evaluation and future directions,” *presented at the Telesystems Conference*, 1992. NTC-92., National, 1992, vol. 13, pp. 15–23.

[23] Carlos R. Aguayo González, Carl B. Dietrich, and Jeffrey H. Reed, “Understanding the Software Communications Architecture,” *IEEE Communications Magazine*, vol. 47, no. 9, pp. 50–57, Sep-2009.

[24] J. Martin and F. Clermidy, “Mobile Digital Baseband: From Configurable to Evolutive Platforms,” in *Mobile and Wireless Communications Summit, 2007. 16th IST*, July, pp. 1–5.

[25] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive Radio Wireless Networks: A Survey," *COMPUTER NETWORKS JOURNAL (ELSEVIER)*, vol. 50, pp. 2127–2159, 2006.

[26] X. Li and S. A. Zekavat, "Spectrum sharing across multiple service providers via cognitive radio nodes," IET Communications, vol. 4, no. 5, pp. 551–561, Mar. 2010.

[27] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," IEEE Communications Surveys Tutorials, vol. 11, no. 1, pp. 116–130, Quarter.

[28] Wipro Technologies, "Software-Defined Radio." broadcastpapers, Aug-2002.

[29] P. A. Bernstein, "Middleware: a model for distributed system services," *Commun. ACM*, vol. 39, no. 2, pp. 86–98, Feb. 1996.

[30] "Common Object Request Broker Architecture (CORBA) Specification, Version 3.2." Object Management Group (OMG), Nov-2011.

[31] "Software Communications Architecture specification Version 2.2.2." JTRS Standards Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS), 15-May-2006.

[32] Wireless@VirginiaTech, "OSSIE- SCA-Based Open Source Software Defined Radio", http://ossie.wireless.vt.edu/

[33] C. R. A. Gonzalez, C. B. Dietrich, S. Sayed, H. I. Volos, J. D. Gaeddert, P. M. Robert, J. H. Reed, and F. E. Kragh, "Open-source SCA-based core framework and rapid development tools enable software-defined radio education and research," IEEE Communications Magazine, vol. 47, no. 10, pp. 48–55, Oct. 2009.

[34] "omniORB – Free High Performance ORB", http://omniorb.sourceforge.net/

[35] J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 4th ed. Addison Wesley, 2007.

[36] Wael A. Murtada, Mohamed M. Zahra, Magdi Fikri, Mohamed I. Yousef, and Salwa El-Ramly, "Impact of the Use of Object Request Broker Middleware for Inter-Component Communication in C6416 DSP Based SCA Radio Systems," *Journal of Computer Science*, vol. 8, no. 6, pp. 957–964, 2012.

[37] Á. P. Navarro, R. Villing, and R. Farrell, "Software Defined Radio architectures evaluation," *in SDR Technical Conference and Product Exposition Washington*, D.C., 2008.

[38] Tiago Rogério Mück and Antônio Augusto Fröhlich, “HYRA: A Software-defined Radio Architecture for Wireless Embedded Systems,” *in the 10th International Conference on Networks*, St. Maarten, The Netherlands Antilles, 2011, pp. 246–251.

[39] L. Dunst, S. Aslam-Mir, B. Duthler, E. Miles, and P. Balister, "Use of novel power control mechanisms in an SCA waveform," *in SDR Technical Conference and Product Exposition*, Washington, D.C., 2009.

[40] F. Berthelot, F. Nouvel, and D. Houzet, “Partial and dynamic reconfiguration of FPGAs: a top down design methodology for an automatic implementation,” *in Parallel and Distributed Processing Symposium., IPDPS 2006*, 2006.

[41] Thomas Schmid, Oussama Sekkat, and Mani B. Srivastava, “An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios,” in WiNTECH`07, Montréal, Québec, Canada, 2007.

[42] George Nychis, Thibaud Hottelier, Zhuocheng Yang, Srinivasan Seshan, and Peter Steenkiste, “Enabling MAC Protocol Implementations on Software-Defined Radios,” *in NSTDI’09*, 2009.

[43] VMWare Inc., “VMWare workstation”, http://www.vmware.com/

[44] Simon S. Lam, “A Carrier Sense Multiple Access Protocol for Local Networks,” Computer Networks: The International Journal of Distributed Informatique, vol. 4, no. 1, pp. 21–32, Feb. 1980.

[45] D. P. Bertsekas and R. G. Gallager, Data networks. Prentice Hall, 1992.

[46] Verhulst, P.-F. "Recherches mathématiques sur la loi d'accroissement de la population." Nouv. mém. de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles 18, 1-41, 1845.

[47] Verhulst, P.-F. "Deuxième mémoire sur la loi d'accroissement de la population." Mém. de l'Academie Royale des Sci., des Lettres et des Beaux-Arts de Belgique 20, 1-32, 1847.

[48] Mathwork Inc., “MATLAB”, http://www.mathworks.com/products/matlab/

[49] S. M. Kim, J. W. Chong, B. H. Jung, M. S. Kang and D. K. Sung, "Energy-Aware Communication Module Selection through ZigBee Paging for Ubiquitous Wearable Computers with Multiple Radio Interfaces," *in Wireless Pervasive Computing 2nd International Symposium*, 2007.

[50] IEEE 802.11 Specification, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 29 March 2012.

[51] IEEE 802.15.1 Specification, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs), 14 June 2005

[52] IEEE 802.15.4 Specification, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), 16 June 2011

[53] Seigneur, J., Titi, X., and El Maliki, T. 2010. Towards mobile/wearable device electrosmog reduction through careful network selection. *In Proceedings of the 1st Augmented Human international Conference,* Megève, France, April 02 - 03, 2010. AH '10. ACM, New York, NY, 1-5.

[54] Roberto de Matos, Tiago Rogério Mück, Antônio Augusto Fröhlich, and Leandro Buss Becker, "Evaluation of PHY Reconfiguration Latency in SDR Gateway for WSN," *in Proceedings of the International Information and Telecommunication Technologies Symposium*, Florianópolis, Brazil, 2009, pp. 199–202.

[55] Péter Völgyesi, Janos Sallai, Sandor Szilvasi, Prabal Dutta, and Akos Ledeczi, "Marmot: A Novel Low-Power Platform for WSNs.," *in Networked Digital Technologies, Part II*, Springer, 2010, pp. 274–280.

[56] B. Gu, J. Jung, K. Kim, J. Heo, N. Park, G. Jeon, and Y. Cho, "SWICOM: An SDR-Based Wireless Communication Gateway for Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 1593–1605, May 2010.

[57] Pasd Putthapipat, Jean H. Andrian, and Chen Liu, "Studies on Inter-component Communication Latency based on Variation Number of Components and Packet size in SDR-SCA Waveform Application," IJCSE (accepted)

VITA

PASD PUTTHAPIPAT

2006	B.Eng., Computer Engineering Assumption University Bangkok, Thailand
2006-2013	Lecturer in Computer Engineering (On study leave) Assumption University Bangkok, Thailand
2012	M.S., Computer Engineering Florida International University Miami, FL
2010-2013	Doctoral Candidate Florida International University Miami, FL

PUBLICATIONS AND PRESENTATIONS

Journal

Pasd Putthapipat, Jean H. Andrian, and Chen Liu, "Studies on Inter-component Communication Latency based on Variation Number of Components and Packet size in SDR-SCA Waveform Application," IJCSE (accepted)

Conference

Pasd Putthapipat, Jean H. Andrian, and Khokiat Kengskool, "Energy Aware Waveform Selection for SDR," *in MAESC 2011*, Memphis, TN, 2011.

Tosmate Cheocherngngarn, Jean H. Andrian, Deng Pan, Khokiat Kengskool, and Pasd Putthapipat, "Fair Bandwidth Allocation Algorithm for Lower Power Consumption on Packet Switch," in MAESC 2011, Memphis, TN, 2011.

X. Jin, P. Putthapipat, D. Pan, N. Pissinou, and S. K. Makki, "Unpredictable Software-based Attestation Solution for node compromise detection in mobile WSN," presented at the GLOBECOM Workshops (GC Wkshps), 2010 IEEE, Miami, FL, 2010.

OTHER ACTIVITIES

Teaching Assistant: Electronics I Laboratory, Integrated Circuits Laboratory

Graduate Assistant:	ABET Curriculum Development Electrical and Computer Engineering department web administrator.