REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

TESTING DETERMINISTIC AVIONICS NETWORKS USING ORTHOGONAL ARRAYS

Muhammed Efe ALTUNTAŞ

MASTER OF SCIENCE THESIS
Department of Avionics Engineering
Program of Avionics Engineering

Supervisor
Prof. Erhan AKDOĞAN

Co-supervisor
Dr. İbrahim HÖKELEK

February, 2022

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

TESTING DETERMINISTIC AVIONICS NETWORKS USING ORTHOGONAL ARRAYS

A thesis submitted by Muhammed Efe ALTUNTAŞ in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 18.02.2022 in Department of Avionics Engineering, Program of Avionics Engineering.

Prof. Erhan AKDOĞAN
Yıldız Technical University
Supervisor

Dr. İbrahim HÖKELEK
TÜBİTAK, BİLGEM
Co-supervisor

Approved By the Examining Committee

Prof. Erhan AKDOĞAN, Supervisor
Yıldız Technical University _____

Assoc. Prof. Aydın YEŞİLDİREK, Member
Yıldız Technical University _____

Asst. Prof. Selçuk CEVHER, Member
Karadeniz Technical University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Testing Deterministic Avionics Networks Using Orthogonal Arrays supervised by my supervisor, Prof. Erhan AKDOĞAN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Muhammed Efe ALTUNTAŞ

Signature

Dedicated to my family

ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis advisor, Prof. Erhan AKDOĞAN, who always supported and helped me. I would like to express my sincere thanks to my thesis co-advisor Dr. İbrahim HÖKELEK, who has always supported me with his help, guidance, and encouragement during my master's degree.

I would like to thank Prof. Umit Uyar and Dr. M. Selim Demir for their contributions throughout the thesis. I would also like to thank my colleagues, who are part of the TAI family, who contributed to the preparation of this thesis.

I would like to express my gratitude to my beloved mother and father, who stood by me with their support and love at every moment and did not spare any sacrifice for me. Also, I would like to express my gratitude to my sisters for their sincere support, sacrifice, and always wishing the best for me.

I would like to express my gratitude to my beloved wife, who has always been with me with her love and support at every moment, and who has always lightened my responsibilities.

I am grateful to my twin babies, who have been a source of morale and motivation for me throughout this tiring and challenging process.

Above all, I thank God for allowing me to finish this challenging process successfully.

Muhammed Efe ALTUNTAŞ

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS viii

LIST OF FIGURES x

LIST OF TABLES xi

ABSTRACT xii

ÖZET xiv

1 INTRODUCTION 1
- 1.1 Literature Review . . . 1
- 1.2 Objective of the Thesis . . . 3
- 1.3 Hypothesis . . . 3
- 1.4 Related Work . . . 4
- 1.5 Organization of the Thesis . . . 7

2 HISTORY OF AVIONICS ARCHITECTURES AND NETWORK PROTOCOLS 8
- 2.1 Federated Avionics Architectures . . . 8
  - 2.1.1 ARINC 429 . . . 9
  - 2.1.2 ARINC 629 . . . 10
  - 2.1.3 MIL-STD-1553 . . . 11
- 2.2 Integrated Modular Avionics (IMA) . . . 12
  - 2.2.1 ARINC 664 - Avionics Deterministic Network . . . 13

3 ORTHOGONAL ARRAYS 21
- 3.1 Same level OAs . . . 21
- 3.2 Mixed level OAs . . . 23

4 TESTING AVIONICS NETWORKS USING ORTHOGONAL ARRAYS 25
- 4.1 Test Space for DTN Switch . . . 25
- 4.2 Complexity of the Test Space . . . 26
- 4.3 Using Chained OAs for DTN Test System . . . 27

5 IMPLEMENTATION OF DTN TEST SYSTEM **31**
5.1 Preparation to Implementation 31
5.1.1 Preparation of Configuration Files 31
5.1.2 Preparation of AIM Test Tool 33
5.2 System Architecture 33
5.3 Current Status of Test System 35

6 RESULTS AND DISCUSSIONS **36**

REFERENCES **37**

PUBLICATIONS FROM THE THESIS **42**

LIST OF SYMBOLS

λ	Coefficient
L_{max}	Maximum Frame Length
L_{min}	Minimum Frame Length
F	Number of Factor
L	Number of Level
P	Number of Ports
N	Number of Runs
S	Number of Strength
OA	Orthogonal Array
p	Priority
$\sum$	Summation Symbol
C	Total Number of Configuration Combinations for a VL

LIST OF ABBREVIATIONS

ARINC	Aeronautical Research Incorporated
AFDX	Avionics Full-Duplex Switched Ethernet
ASN	Ancillary Sensor Network
BE	Best Effort
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
CRC	Cylic Redundancy Check
DAL	Design Assurance Level
DTN	Deterministic Network
DUT	Device Under Test
ES	End System
FCS	Frame Check Sequence
Gbps	Gigabit Per Second
IMA	Integrated Modular Avionics
I/O	Input/Output
kbps	Kilobit Per Seconds
LRU	Line Replicable Unit
LRM	Line Replicable Module
kbps	Megabit Per Seconds
max	Maximum
MIB	Management Information Base
min	Minimum

MIL-STD-1553	Military Standart 1553
ms	Millisecond
NC	Network Calculus
OA	Orthogonal Array
OS	Operating System
PC	Personal Computer
PCI-e	Peripheral Componenet Interconnect - express
PXI	PCI eXtension Instrumentation
SNMP	Simple Network Management Protocol
SoC	System on a Chip
SUT	Switch Under Test
SW	Switch
UDP	User Datagram Protocol
us	Microsecond
VL	Virtual Link

LIST OF FIGURES

Figure 1.1 Design assurance level 2
Figure 2.1 Sample federated avionic architecture 9
Figure 2.2 Structure of ARINC 429 10
Figure 2.3 Federated avionics architecture with ARINC 629 bus 11
Figure 2.4 Sample MIL-STD-1553 bus 11
Figure 2.5 Federated and IMA modules architecture 13
Figure 2.6 An example SW DTN network topology 14
Figure 2.7 Internal structure of switch 15
Figure 2.8 Internal structure of end system 18
Figure 2.9 Traffic regulation by end system transmitter 19
Figure 5.1 Parameters of oapackage python script 32
Figure 5.2 Architecture and information flow of DTN test system 32
Figure 5.3 Architecture and information flow of DTN test system 34
Figure 5.4 GUI sample for real-time status monitoring of DTN test system . . 35

LIST OF TABLES

Table 3.1 Example for same level OA: an experiment with 3 factors each with 2 levels 22
Table 3.2 Example for same level OA: runs for $OA(4,3,2,2)$ 23
Table 3.3 Example runs for a mixed level OA=$(12, 2^4 3^1, 2)$ 24
Table 4.1 Sample configuration parameters for DTN switch 27
Table 4.2 DTN switch variables for OAs: the number of incoming VLs to ports for OA_{g1} 29
Table 4.3 DTN switch variables for OAs: test parameters for VLs for $OA_{i,j}$ . . . 29

ABSTRACT

Testing Deterministic Avionics Networks Using Orthogonal Arrays

Muhammed Efe ALTUNTAŞ

Department of Avionics Engineering
Master of Science Thesis

Supervisor: Prof. Erhan AKDOĞAN
Co-supervisor: Dr. İbrahim HÖKELEK

The Integrated Modular Avionics (IMA) concept has become popular to meet the increasing requirements of next-generation aircraft. Ethernet-based deterministic network (DTN) protocols have been developed for the IMA concept. ARINC 664 which is DTN protocol, provides end-to-end communication with bounded latency and jitter for safety-critical applications in avionics systems. Existing certification requirements for equipment to be used on aircraft determine the design assurance levels (DAL) based on the equipment's criticality levels. Due to the safety criticality in the DTN, it is characterized as DAL A, which means the highest criticality. Therefore, a test bed should be prepared that includes all possible scenarios when using a deterministic network. However, It is not achievable to generate a test suite with a feasible size without employing a formal design of experiment method such as orthogonal arrays (OAs) if we consider all possible combinations of DTN switch configuration parameters such as the number of virtual links (VLs), the frame size, and bandwidth allocation gap for each connection. Because the basic logic of the orthogonal array is to obtain all pairwise combinations once and obtain a subsystem that will cover the entire test space. Using OAs is not sufficient to generate test cases representing real-life usage of DTN switches since many combinations of VL parameters may be inadvertently eliminated by the OA generation process. In this thesis, we present an approach, called *chained OAs*, where multiple applications of OAs are applied to the test factors to generate test cases that are representative of real-life usage of DTN switches. The test cases are converted into executable test scripts to be run by the commercial test system.

This study shows that chained OAs method is capable of reducing prohibitively large number of tests required for complex systems, while covering pair-wise combinations of all test parameters.

Keywords: Orthogonal array, test generation, deterministic network, ARINC 664, avionics

ÖZET

Ortogonal Diziler Kullanılarak Deterministik Ağların Test Edilmesi

Muhammed Efe ALTUNTAŞ

Aviyonik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Prof. Dr. Erhan AKDOĞAN
Eş-Danışman: Dr. İbrahim HÖKELEK

ARINC 664, aviyonik sistemlerdeki kritik güvenlik uygulamaları için sınırlı gecikme ve titreşim sağlayan, yaygın olarak kullanılan Ethernet tabanlı bir deterministik ağ (DTN) standardıdır. Deterministik ağ kullanımı açısından kritik görevlere sahip olması nedeniyle güvenlik değerlendirmesi DAL A olmalıdır. Bu nedenle deterministik ağ ekipmanlarının tüm olasılıkları kapsayan çok ciddi testlere tabi tutulması gerekmektedir. Her bağlantı için sanal bağ, paket boyutu, bant genişliği tahsis protokolü gibi bütün ağ anahtarı konfigürasyon parametreleri ele alındığında tüm olasılıkları kapsayan bir test süreci düşünülemez. Çünkü bu durum hem zaman hem de maliyet açısından uygulanabilir değildir. Dolayısıyla test uzayını anlamlı bir şekilde küçülterek bütün uzayı kapsayacak bir alt uzay elde etmek gerekir. Bu özelliklere sahip uygulanabilir bir test düzeneği elde edilmesine yardımcı olacak deney tasarımı metodunun adı ortogonal dizilerdir. Ortogonal arraydeki temel amaç, bütün deney tasarımı parametrelerinin ikili kombinasyonlarını çıkartarak bütün olasılıkları elde etmektir. Fakat, ortogonal dizi üretim sürecinde sanal bağ parametrelerinin birçok kombinasyonu yok sayılabilir. Bu nedenle ortogonal diziler, deterministik anahtarların gerçek hayattaki kullanımını temsil eden test senaryolarını oluşturmak için tek başına yeterli değildir. Bu tezde deterministik ağ anahtarlarının gerçek hayattaki kullanımını temsil eden test senaryoları oluşturmak için test faktörlerine ardışık bir şekilde ortogonal dizileri uygulayarak zincirleme ortogonal dizi adında yeni bir yaklaşım sunuyoruz. Sonuç olarak zincirleme ortogonal dizilerden elde edilen test senaryoları, ticari bir test sistemi tarafından çalıştırılmak üzere yürütülebilir test

komut dosyalarına dönüştürülür. Bu çalışma, zincirleme ortogonal diziler yönteminin, tüm test parametrelerinin ikili kombinasyonlarını kapsarken, karmaşık sistemler için gereken çok sayıda testi azaltabildiğini göstermektedir.

Anahtar Kelimeler: Ortogonal diziler, test üretimi, deterministik ağlar, ARINC 664, aviyonik

1
INTRODUCTION

1.1 Literature Review

Over time, there have been significant developments in aircraft systems. Avionics systems have also taken their share from this development. The avionics architectures used in the early examples did not work as a system but as a separate module. For this reason, point-to-point and limited-speed communication met the needs of avionic modules. Therefore, it was not difficult to provide the requirements such as low jitter, low packet loss, and low latency, which must meet in terms of determinism. However, the development of digital electronics has led to an increase in avionic systems and tasks. As a result, these developments have caused to development of different and complex avionics architectures. Due to changing communication requirements, complex avionics architectures required new protocols. Therefore, various complex communication protocols are emerged such as ARINC 429, MIL-STD-1553, and ARINC 629. The primary purpose of these protocols is to provide a deterministic network. Therefore, these various protocols have used it to provide low packet loss, low latency, and jitter. These developments have gained another dimension with the IMA architecture used today. Because the ARINC 664 protocol, which is used thanks to this architecture, creates a network far beyond just a data bus.

The ARINC 664 protocol is a virtual link-based protocol. Virtual links are connections established between end systems. Traffic is provided by the frames sent from these links. In order to obtain deterministic traffic, it is necessary to keep the packet loss, jitter, and latency to the lowest level in the frames sent over virtual links. Therefore, generated traffic must have precomputed and predictable packet loss jitter and latency. In this case, network traffic is regulated offline by constrained different parameters of virtual links such as BAG, L_{min}, L_{max}, and jitter. Therefore, due to the regulated traffic, ARINC 664 was named as a deterministic network. The ARINC 664 network can be used in an avionic sub-system or an entire network application in the aircraft. This case shows that the IMA architecture is a network-centric architecture and how important ARINC 664 is for the architecture. Therefore, ARINC 664 must-have DAL

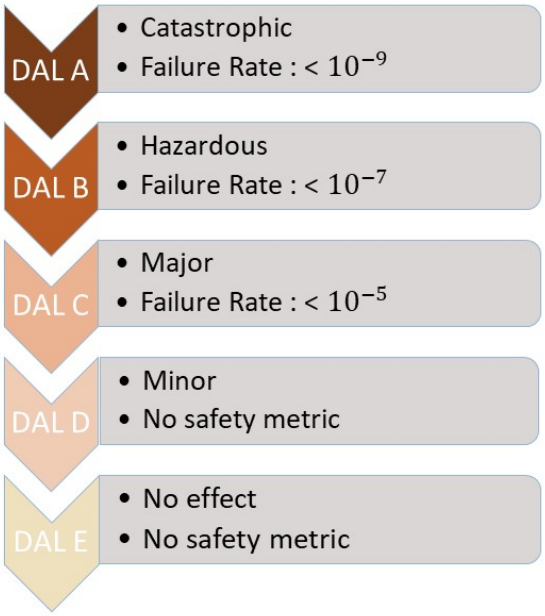


Figure 1.1. Design assurance level

A level reliability in a design for the IMA architecture [1]. Design Assurance Level A (DAL A) means serious testing of the relevant equipment.

Certifications must be obtained from the necessary authorities to use the avionic systems in the aircraft. These certifications can be achieved at the module level in the airliner. An ARINC 664 network module has safety requirements for using in an aircraft. These requirements determine whether the equipment should be certified and the level of criticality. According to the task the equipment undertakes, this critical level is classified according to the requirements in the classification table called the design assurance level (DAL) which is determined the classification of criticality of the level. There are five different DAL levels, as seen in Figure 1.1 [2]. These are in order the most important error level, which should be the lowest error rate, to the unimportant error levels. First of all, the DAL level A is for catastrophic conditions. In other words, the critical systems developed have an error rate less than 10^{-9}. Therefore, systems that will result in accidental breakdown and death in the event of an error in the aircraft are considered DAL A. The DAL level that causes the conditions called Hazardous is DAL B. Systems developed at this DAL level should have an error rate of 10^{-7} or less. When they make a mistake in these systems, they may suffer an accident at a level that cannot be called specific, like DAL A, and situations that may result in death occur. At this point, the brake systems of the aircraft can be given as

an example. DAL C, on the other hand, has an accident rate of 10^{-5} or less, as seen in Figure 1.1. Also, accidents with DAL C levels are called major accidents. We can give an example of the breakdown of the backup system in the aircraft for DAL C. The DAL D level, the minor level, is the situation that can only make flight difficult. An example is that ground navigation systems make mistakes. Finally, DAL E is the passenger entertainment systems' fault that does not affect the flight. There was no need to give any failure rate to DAL D and DAL E [3], [4].

1.2 Objective of the Thesis

When ARINC 664 is considered in terms of DAL levels, it needs to work with DAL A, that is, an error rate of one billionth, to operate safely in the aircraft. Therefore, it should be designed with an accuracy that will make errors at the determined rates because the ARINC 664 protocol is a network designed for significant frames that are time-critical in the aircraft and should not drop. It is necessary to prove that ARINC 664 equipment, which has this criticality and error rate, is designed and working correctly. For this reason, these network elements must pass serious tests. These tests should cover all the possibilities that the equipment may encounter, both in-ground position and while flying through the air. It is impossible to apply tests that will include all possibilities directly, both in terms of time and cost. Experimenting with all possibilities means creating configuration files for ARINC 664 equipment containing all possibilities. Because all traffic scenarios are implemented thanks to the configuration files that are prepared and uploaded to the network. At this point, when you look at the switch and end system variables described in Section 4.1 and the values they can take, you can see how huge our configuration space is. As a result, it is not possible to apply tests that will include all possibilities directly, both in terms of time and cost. Therefore, with orthogonal arrays, a scientific method, the test space was reduced to cover the entire test space. As a result, orthogonal arrays provided a significant gain in both time and cost.

1.3 Hypothesis

Ethernet-based deterministic network (DTN) technologies have been developed to cope with the high speed data communication requirements of the modern avionics applications. ARINC 664, which is a commonly used DTN technology, provides the bounded latency and jitter when safety critical applications communicate with each other over the avionics networks within an aircraft. Because of their DAL A safety-critical nature, the availability of large number of configuration options and high speed operation using thousands of virtual links, testing ARINC 664 switches is

a challenging task.

Prohibitively great numbers of combinations of DTN configurations necessitates that formal test generation techniques be employed to effectively reduce the number of tests. Orthogonal array (OA) is a popular technique to optimize test suite size for systems with many features while maintaining their coverage [5–11]. However, using a single level OAs is not sufficient to generate test cases representing realistic operating conditions of DTN switches. Many combinations of switch configuration parameters may be inadvertently eliminated by the OA optimization process. In this thesis, we present an approach, called *chained OAs*, where multiple and consecutive applications of OAs are employed to generate tests that are representative of real-life usage of DTN switches. Chained OAs method allows for preserving the number of combinations for a set of selected features, while eliminating redundant combinations for others as decided by test engineers. We define two stages for the chained OA method in test generation of DTN switches. First, combinations of different number of virtual links (VLs) that can be assigned to each switch port are generated by OAs. Then, the OA output elements are used as inputs to a set of second stage OAs to determine the remaining switch parameters such as transmission time gaps between consecutive frames, minimum and maximum frame sizes, and VL priorities. The OA outputs are generated using an open source Python library, called *oapackage* [12]. The tests generated by the chained OAs method are converted into executable scripts, which are applied to a DTN switch under test by a commercial test system, called AIM [13].

1.4 Related Work

ARINC 664 equipment has been designed many times, and various testing activities have been carried out to prove their correct operation. While some of these tests only include performance tests, some appear as functional validation tests. In this subsection, the testing activities of ARINC 664 equipment prepared so far are mentioned.

One of the most crucial performance tests for ARINC 664 equipment is the end-to-end measured latency tests. Chen and Song [14] aimed to measure the end-to-end delay by performing a loopback test in their study. They managed to do this by throwing a package from the source end system to the destination end system and throwing the same package from the destination to the source end system, and making the necessary measurements. In addition, they have made functional tests by injecting various errors such as VL error, port error, network error into the packets. While conducting these tests, they used an industrial PC with an AFDX end system, AFDX switch, and AFDX

end system they developed.

In another study [15], the study is contained that implementing the AFDX end system on a Zynq SOC. In the setup, there is one cisco Ethernet switch, one TTech AFDX switch, two end systems developed by authors and one end system developed by CES. This test system first checked the compatibility of the packets thrown from the end system they designed with the protocol. While doing this, they used the Wireshark program. They also used CES CAD-X for more detailed investigations. They also had the opportunity to compare the data they observed in Wireshark with the CES CAD-X data. Latency tests were also performed using CES CAD-X. The latency test was done by sending multiple frames in a window time slot and then checking how long it took the hardware to send a frame. CAD-X displays the time between the acquisition of two successive frames testing different frame lengths. They also performed various hardware sampling and queuing tests.

Qu, Yang et al. [16] established a PXI Bus-based test system in their study. This test system includes one PXI-C664 AFDX simulation card, one PXI-8115 controller, and AFDX equipment to be tested. In addition, according to the test requested from the Ethernet interface on the PXI-8115 controller, the data required for generating error-injected or standard error-free packages are sent to the PXI-C664 AFDX simulation card via PXI Bus. In addition, the behavior of AFDX equipment and PXI-C664 equipment is monitored over this PXI Bus. In this test setup, they inject error cases such as VL ID error injection, UDP port number error, fault of IP address, fault of AFDX payload, sequence number error into the packets and send the generated error packets to the AFDX equipment. They examine the behavior of AFDX equipment against them.

In another related recent study [17], the authors developed a test system consisting of two different equipments. The first equipment is used for the network configuration tool while the second equipment is used for the avionics bus test tool. The network configuration tool generates configuration files in the form of .xml or .hex used for configuring end systems. The avionics bus test tool monitors test transmission accuracy and ARINC 664 network speed. It also does various VL tests and loopback tests. Again in this study, PXI-C664 AFDX simulation card and PXI-8115 controller were used.

Reliability is essential for avionic networks. Li, Huang, and Liao [18] were studied on test profile design for reliability testing of ARINC 664 networks. They focused on traffic-based reliability testing. Because the testing since the performance reliability depends on the traffic load of the network system. If the traffic load is high, the

transmission latency and error rate will be increased. Therefore, ARINC 664 networks should be tested under different traffic profiles. VL ID, message size and frequency, and traffic duration parameters are defined for creating different traffic loads. The traffic load was determined by arranging these parameters according to different scenarios.

Alena and others [19] creates test-bed for AFDX Ancillary Sensor Network (ASN). A test infrastructure was created to test various AFDX commercial products to perform performance and compatibility tests. A representative ASN configuration was used for this test engine. The primary purpose of this test-bed is to test time determinism and network throughput features of AFDX network elements. Error situations were determined by creating a heavy traffic load on the network. There were two requirements to perform the test cases. The first was to create a test vector set containing parameters such as frame transmission frequency BAG, packet data sizes, and VL numbers. Another requirement is to analyze while using the 100 Mbps bus at full capacity. The software that will determine the traffic flow of the system, generate messages, and inject errors into the messages it produces, has been written by the team. In addition, the test team has developed their software to meet various needs such as jitter analysis, bandwidth analysis, error conditions, and latency analysis for monitoring tasks and analysis. By bringing them all together, they tested various AFDX network elements in the system they established. They used program called Panorama V to analyze the statistical results from their software.

In different ARINC 664 test study [20], ARINC 664 protocol conformance test is applied to designed ARINC 664 switch with autotest platform. The reason for the switch selection is that it is the core element of ARINC 664 networks. The test cases are divided into two groups. These are functional and performance tests. The functional tests include communication, filtering, and policing. Also, the performance tests include wire-speed tests for receiving and policing, latency, and jitter tests. These tests are applied with three different test methods. One of them is the concurrent test subset. Sending traffic flow concurrently to all switch input ports is needed for switching and performance tests. Another subset is the injection traffic error test subset. This test subset is important to understand switch actions against the negative case of the protocol. This subset includes filtering and policing functions. Frames, which do not match with the features of ARINC 664 protocol, should be dropped. That's why, negative cases should be needed to test for conformance tests. The last test subset is integrity. It is applied to test the other features of the switch such as pin programming, monitor, configuration, which must be complied with the ARINC 664 protocol. The architecture of the test system consisted of end systems, embedded test manager, and remote controller. This remote test manager is connected to an embedded test CPU. When tests run from remote tests manager, embedded test

CPU sends configurations to prepare the end system on the PCI bus and create all traffic with configuration. Then, it runs the test case. Lastly, it reports the results of the test case. This test platform tests the switches which have 8 to 24 ports. This autotest platform has different test scenarios concerning switch features. There are 120 different test cases. It provides hundred percent of coverage of the switch features.

One of test and simulation module manufacturer who is AIM Gmbh [21], designs the ARINC 664 test-bed for the switch and end system of the protocol. This test-bed consists of two different elements. These are test manager software and test hardware. This test-bed has different specific test scenarios for testing ARINC 664 features of modules. For switch, it has more than 59 tests in ten different test headings such as filtering, policing, switching, performance, monitoring, configuration etc. Similar test cases exist for end system tests [22]. The main idea of the AIM ARINC 664 test tool is that it is to be able to run functional and performance tests. For each test, traffic is generated from configurations. Tests are performed by sending the generated packets to the DUT.

To conclude, various test environments have been developed for ARINC 664 equipment in various studies. Performance and functional tests were generally performed in these test environments. The majority of these tests are based on measuring the end-to-end delay, injecting the error, and observing the behavior of the systems against various error conditions. Thus, it has been understood whether the performance and functional requirements of the systems are fully met.

1.5 Organization of the Thesis

This thesis is organized as follows. The main purpose in Section 2 is to show that the development of avionics architectures leads to the emergence of new network protocols. As a result of this, it is emphasized that the IMA architecture used today plays an important role in the emergence of the ARINC 664 protocol. Especially, ARINC 664 avionics deterministic networks are outlined in Section 2.2.1.The fundamental features of OAs are introduced in Section 3. Section 4 presents the complexity of test space for avionics deterministic networks and impact of the chained OA method to reduce the test suite size while preserving important test features. The architecture of our DTN switch test system is summarized in Section 5.

2
HISTORY OF AVIONICS ARCHITECTURES AND NETWORK PROTOCOLS

With the development of technology over time, there have been revolutionary developments in airplanes. When the first airplanes started to be used, they consisted of mechanical systems. Especially, in the second world war, when it became clear that air supremacy was more important, the studies in this field gained momentum. The first use of electronics in aircrafts was analog electronic systems between 1950 and 1960. However, the development that can be considered vital for aircraft is seen as using different electronic devices and systems that emerged with transistors in aircraft. With the rapid development of digital electronics, electronic systems found a severe place in aircraft systems after the 1970s [23]. Thus, a separate sub-branch, called avionics, emerged from the meeting of electronics with aviation. This situation reveals the importance of electronics in aircraft systems.

The effects of the changes in the avionics architecture topology also determined the changes in the avionics network. Therefore, over time, the needs of different architectures led to the emergence of new network protocols. The last point these network protocols have reached today is ARINC 664. This section will examine the architectures and network protocols that emerged in the process up to ARINC 664.

2.1 Federated Avionics Architectures

Federated avionics architectures consist of different avionic modules called Line Replaceable Unit (LRU) or Line Replaceable Module (LRM) [24]. Each LRU is an electronic board. On the other hand, these electronic cards are in a structure that can undertake only one task. The reason for this can be seen in the structure of the federated modules in Figure 2.1. Each module has its own I \O pin, CPU, infrastructure OS, and application software in federated architectures [25]. As seen in Figure 2.1, only one application software is run on the CPU in the module. Therefore, all modules are designed to perform only one task. That's why, the number of used avionics can

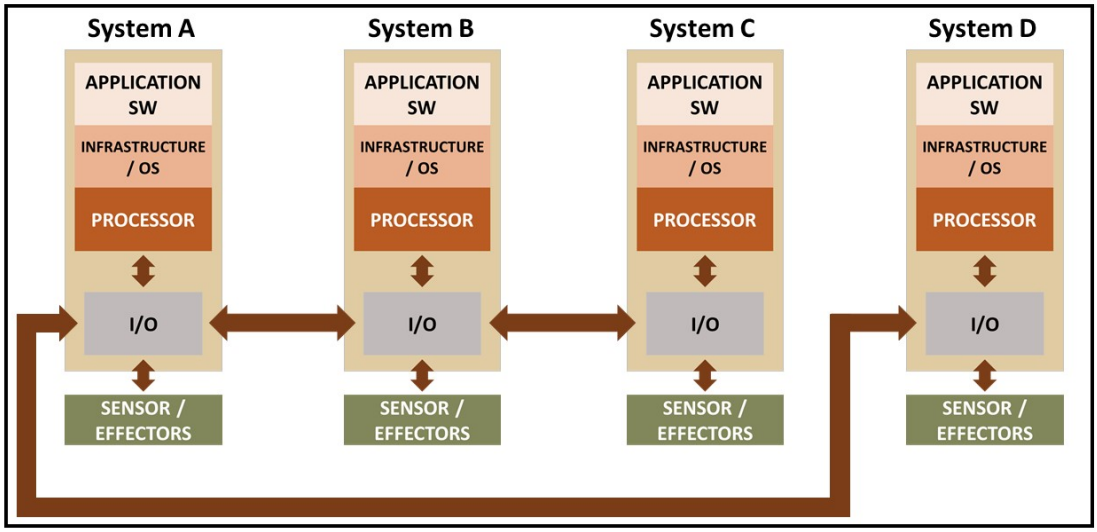


Figure 2.1. Sample federated avionic architecture

be higher than number of avionics of IMA architecture or any other architecture. This situation can create dependency on different companies in terms of COTS products. In addition, aircraft weight increases in terms of the number of avionics used and cabling. This is an undesirable situation in the design of aircraft.

Federated architectures that the combination of different modules will form need a protocol that will enable communication between modules. As seen in Figure 2.3, larger topologies are achieved at the system level by using the ARINC 429, ARINC 629, or MIL-STD-1553 protocol together with federated avionics architectures [26]. In federated architectures, it is possible for a system consisting of LRUs to communicate with different systems with a different databus over one LRU, as seen in Figure 2.3. This means that the architecture works holistically.

2.1.1 ARINC 429

The ARINC 429 protocol was developed by Aeronautical Radio, Incorporated (ARINC) in 1977 and was finalized in 1983 for the communication of digital electronic systems in aircraft with each other [23] [27]. This communication protocol provides a bidirectional data bus between digital systems in a distributed architecture which is shown in Figure 2.2 [28]. One source (transmitter) and up to 20 sinks (receivers) can be connected to this data bus [29]. The maximum data rate that can be provided between these connected devices is 100 kbps [30]. This is the high data rate required for the digital systems of that day. Also, ARINC 429 provides a redundant connection between terminals. This shows that ARINC 429 considers the reliability of the

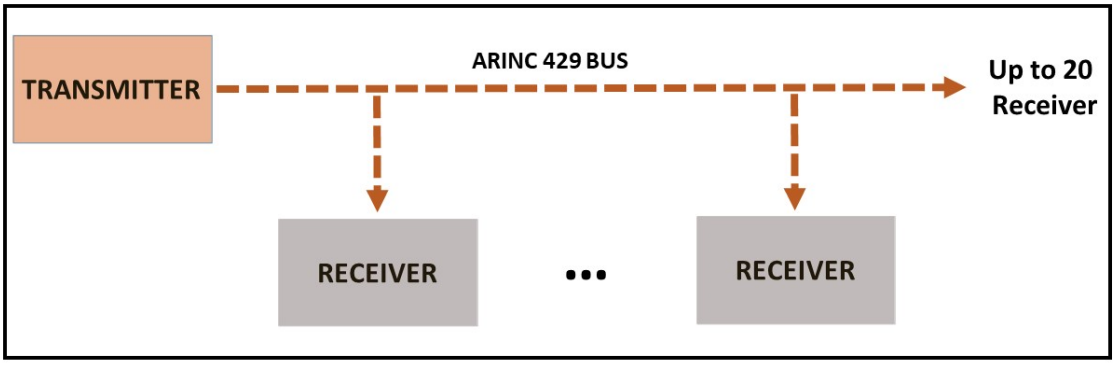


Figure 2.2. Structure of ARINC 429

network.

For federated architectures, the network in Figure 2.3 may be similar to ARINC 429. However, the unidirectional nature of ARINC 429 does not provide inter-system communication which is explained in Section 2.1.2. In addition, the low data rate of ARINC 429 and including less sink modules than ARINC 629 in the protocol shows that it cannot create a topology as large as ARINC 629.

2.1.2 ARINC 629

With federated systems, faster and more complex data paths were needed. One of these needed data paths is the ARINC 629 protocol, which has been used in civil aviation since 1995, developed by Aeronautical Radio, Incorporated (ARINC). This protocol is firstly used in B777 aircraft [27]. ARINC 629 is used for federated systems as seen in Figure 2.3.

Main starting point of ARINC 629 is to exchange data between avionic systems. That's why, acts as bidirectional unlike ARINC 429. In Figure 2.3, there are different ARINC 629 data bus which are supporting up to 120 terminals [31]. Some LRU can connect to more than one data bus. That's why, different subsystems can communicate over the common LRU's. ARINC 629 can host more than one master, namely ARINC 429 mutually sourced terminals. This is a requirement of the previously mentioned bidirectional structure. In order to prevent the packets from being exposed to any collisions in bidirectional structure, a network protocol called CSMA/CA is used to check whether the channel to which the data will be sent is empty [32]. In addition, ARINC 629 supports 2 Mbps data transfer rate, providing 20 times the speed of ARINC 429 [27].

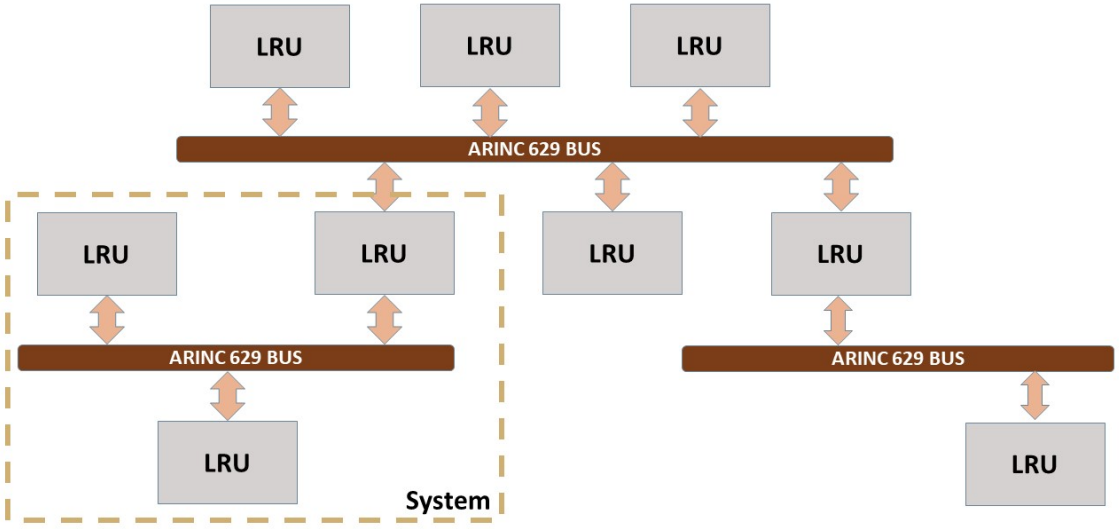


Figure 2.3. Federated avionics architecture with ARINC 629 bus

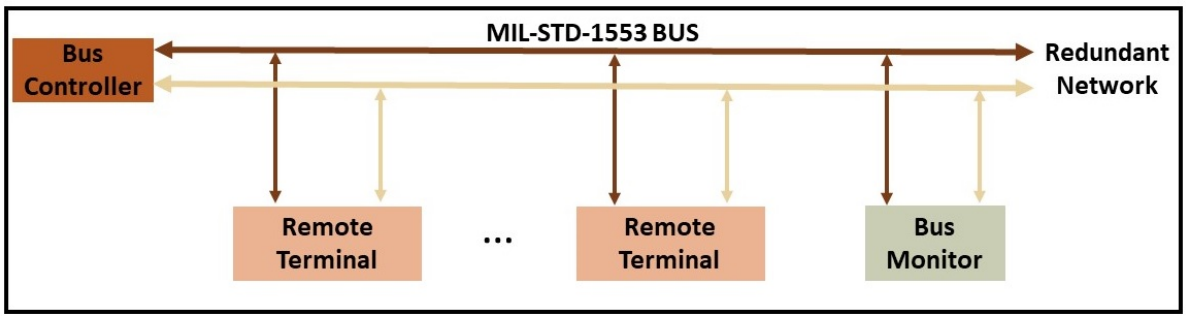


Figure 2.4. Sample MIL-STD-1553 bus

From a safety point of view, ARINC 629 is a protocol that can be used in a redundant structure. ARINC 629 protocol works in a redundant structure against errors on a single data bus. Depending on the structure in the aircraft, each LRU can be connected to more than one ARINC 629 data bus, one of which is redundant to the other.

2.1.3 MIL-STD-1553

Another bus protocol used with federated systems is the MIL-STD-1553 protocol developed by the US department of defense for military systems. It was first tested on F16 aircraft [33]. It has also been used in many sea and land vehicles. It is so widely used because it is simple, deterministic, and reliable. As seen in Figure 2.4, MIL-STD 1553 protocol works in a redundant structure against errors that may occur on a single data bus. Also, it has a low error rate of 1-word fault per 10 million words. This system consists of 3 different components; these components are Bus controller, remote terminal, and bus monitor, which are shown in Figure 2.4 [34]. We can say

that the bus controller a kind of task computer. On the other hand, a remote terminal is a subsystem that acts with the data coming from this task computer. The bus monitor is the component that monitors the bus. MIL-STD-1553 protocol, which can receive 30 remote controls on a line, works at 1 Mbps as half-duplex [35].

2.2 Integrated Modular Avionics (IMA)

IMA technology is a form of avionics architecture that has been used since the 1990s. Although it was first seen in the F22 raptor aircraft in the military sense, it is also used in the B787 and A380 which are civil aviation aircraft [36]. In general, it works with the opposite logic to the federated architecture. It is an architecture based on powerful processors that can run multiple independently software applications. In other words, the main purpose is based on the fact that a piece of hardware can be used for more than one application [36]. As seen in Figure 2.5, the processor and infrastructure operating system used in IMA architecture are common, but more than one independent application is running on the upper layer. This means IMA hardware to do multiple LRU jobs. Due to less hardware usage. there is a significant saving in terms of aircraft weight, size, power consumption [37].

Since the same hardware can run more than one software, it can perform more than one task at the same time. The tasks set for the hardware also determine the application softwares to run. This indicates that the hardware has software-based modularity. In this way, since the updates on the system will be on the software side, serious convenience is provided. The convenience obtained in this development causes a significant reduction in terms of both calendar and cost. In addition, this situation contributes to the redundant structure in the system. A redundant structure is the operation of the same systems independently. Therefore, thanks to this modularity, redundant structures can be easily created [38].

All functions in IMA are connected by common data network. In this way, all avionics in the system can communicate. Nevertheless, the avionics system needs real-time processing. In addition, considering the data sharing situation of critical systems with each other, packet loss, delay, and jitter should be absent or minimized. Also, there is a need for a communication protocol with a wide bandwidth among the developing avionics systems, which is increasing in aircraft. In order to meet these requirements, the system needs a deterministic network. In today's IMA systems, this deterministic network need is met with the ARINC 664 protocol [23].

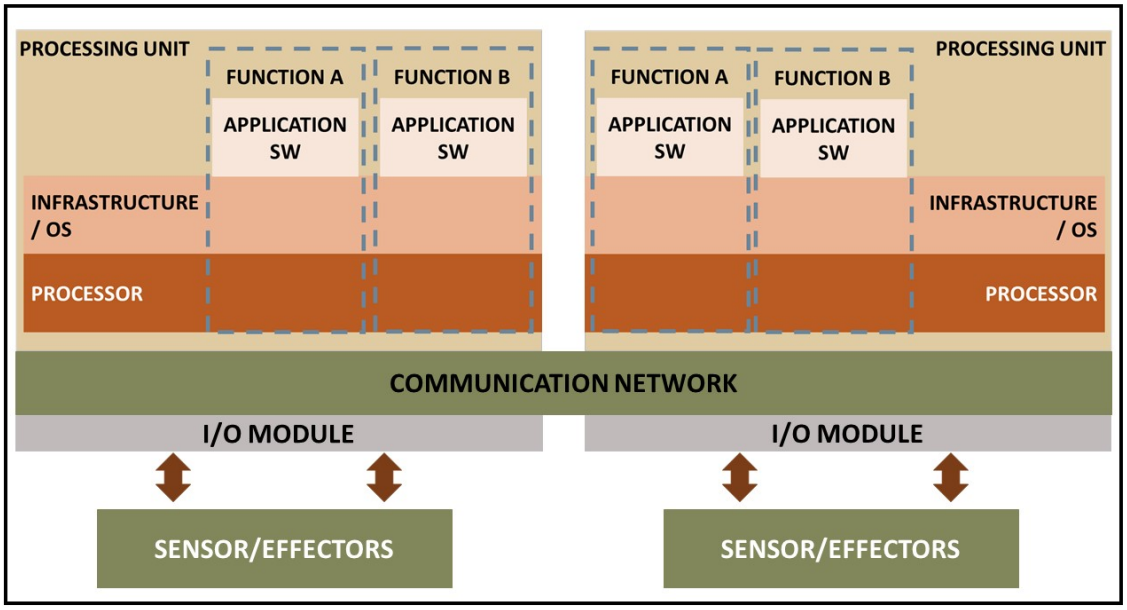


Figure 2.5. Federated and IMA modules architecture

2.2.1 ARINC 664 - Avionics Deterministic Network

The most important elements of an SW DTN include switch (SW), end systems (ESs), and virtual links (VLs) [39]. Applications running on different ESs communicate with each other using VLs that are routed by SW. The bounded latency and jitter requirements for VLs are met by offline planning of SW: the number of VLs, the source and destination ESs, the minimum and maximum frame sizes, and bandwidth allocation gap (BAG) for all VLs [40–42]. The main idea behind providing deterministic communication is to limit the amount of traffic injected by each VL to the network by setting allowed frame size and BAG values, to achieve the desired delay and jitter objectives [43]. BAG specifies the minimum transmission time between the first bits of two consecutive frames. A traffic shaping mechanism running over each VL will ensure that the amount of traffic injected by each node will be controlled and the traffic exceeding the average and peak rates will be shaped. The switch will enforce filtering and policing requirements, limiting fault propagation, and ensuring determinism.

In the ARINC 664 redundancy management architecture, the same frame is transmitted over two independent SWs and one of the received copies is sent to the application layer at the receiver. ARINC 664 does not provide a reliable transmission guarantee that all frames will be delivered to their destinations. However, in a properly planned network, the amount of congestion can be predicted and limited during the planning phase and the priority concept along with the redundancy management provide a reliable data transmission service. For the safety certification of ARINC

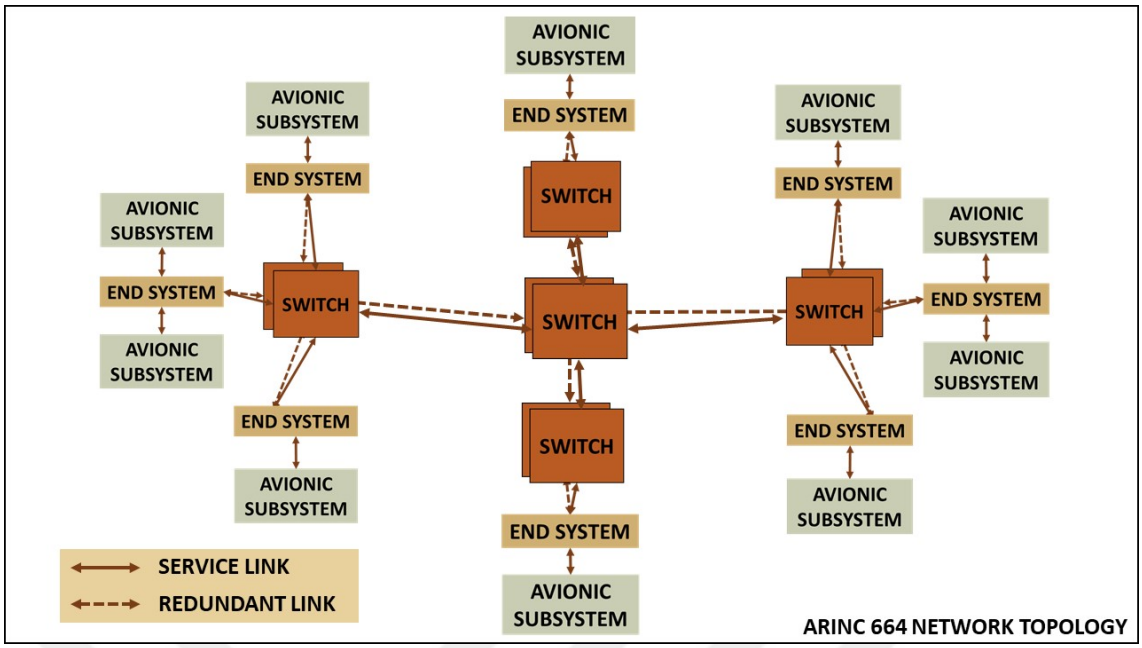


Figure 2.6. An example SW DTN network topology

664, it is a common practice that the network calculus (NC) framework can be applied to calculate the worst-case delay and jitter bounds for a given network configuration [44][45]. Figure 2.6 depicts an DTN network with cascaded SWs,and ESs, each of which is connected to the SWs via service and redundant VLs.

2.2.1.1 Virtual Link

ARINC 664 is a unidirectional protocol built on a virtual link. The virtual link is an ARINC 664 element that connects end systems in the network to send frames to each other [46]. This connection between end systems can be defined as unicast or multicast. An end system can send packets to one or more end systems. Some specific configuration values are defined to the VLs to ensure determinism in the ARINC 664 system. Among these values, the minimum length (L_{min}) that the package can take, the maximum length (L_{max}) that the package can take, BAG (Bandwidth Allocation Gap), jitter values [47]. L_{min} and L_{max} values indicate the range of values that the packet length with the corresponding VL number can take [48]. In this way, the maximum technological latency values that the package can create are predictable. The BAG value is the value that indicates the frequency of sending the packet with the relevant VL. Jitter is the deviation value that can be between the sending frequency of the packet. As a result, virtual links are elements that create a deterministic traffic flow thanks to the configuration values they receive. There is no packet loss in this way, and it is minimized. Latency values are predictable [1].

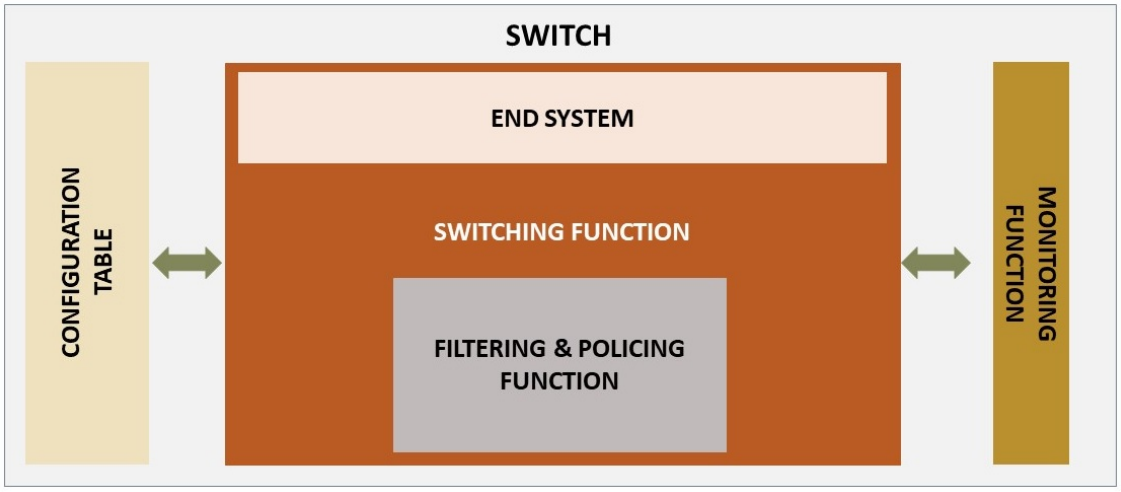


Figure 2.7. Internal structure of switch

2.2.1.2 Switch

ARINC 664 Switch is equipment that directs traffic flowing by end systems or other switches. This equipment has various functional blocks within itself, shown in Figure 2.7. These blocks are policing, filtering, switching, configuration table, and monitoring functions [49] [50]. The mentioned blocks provide the necessary checks of the incoming packets for the ARINC 664 switch to implement the ARINC 664 protocol properly. As a result, the package is dropped if the incoming packages do not come with the desired configurations and features. Otherwise, the packet is forwarded and exits the relevant port. In addition, the end system in the switch is used to receive the packets coming to the switch itself and to send the packets to be sent by the switch. Detailed information about policing, filtering, switching, configuration table, monitoring functions will be given in subsections.

2.2.1.2.1 Filtering Function

The ARINC 664 protocol is an Ethernet-based protocol [51] [52]. It performs all the packet filtering operations required by the Ethernet protocol. For example, Frame Check Sequence (FCS) drops bad packets, packets with alignment errors, and packets that do not match Ethernet's packet size and structure without routing. However, we also need to implement various filtering requirements that are the requirements of the ARINC 664 protocol. The filtering module these operations. The first of these operations is the package length in accordance with the VL parameters described in Section 2.2.1.1. None of the incoming packets should be smaller than the minimum packet size (L_{min}) of the corresponding VL number and larger than the maximum packet size (L_{max}). Anything outside of this range will be dropped directly. Another filtering requirement is whether the incoming packets comply with the ARINC 664

MAC address requirement. The ARINC 664 MAC address must begin at 03:00:00:00, and the first 32 bits must be fixed. The last 16 bits must represent the VL number [40]. While checking the MAC address, it also checks whether the relevant VL is valid. If the related VL is not valid, it drops the packet because it does not have a destination address to forward that VL to. As a result, the filtering module checks whether the incoming packet is transferred without error and whether it is compatible with the ARINC 664 protocol.

2.2.1.2.2 Policing Function

The module that runs the algorithms in which the VL-based traffic generated by the end systems is controlled, is defined as policing. As stated earlier in section 2.2.1.1, various VL parameters make the ARINC 664 protocol deterministic. These are the most critical parameters that determine the traffic flow and rate. Although it is desired to obtain a regular traffic flow with VL parameters, unpredictable or more than predicted jitter or latency situations may occur between the end system and the switch in the network. Also, the traffic flow may be disrupted due to an error situation on the end system side. This may cause the frames to arrive earlier than the determined BAG value. For these reasons, a token bucket algorithm regulates the traffic flow in the switch. With this algorithm, packets arriving before the time interval determined by the BAG value are dropped. In this way, a guaranteed bandwidth value reserved for the relevant VL is provided. Thus, a reliable and deterministic network is obtained by reducing the incoming frames different from the values specified in the configuration and disrupting the traffic flow.

2.2.1.2.3 Switching Function

Switching is the most basic task of a developed switch, is done according to the previously loaded VL table in ARINC 664 switches. In this VL configuration table, there are destination and source addresses for the related VLs. As stated in Section 2.2.1.2.1, the destination port is determined from this table after the source port control of the related VL is passed. According to this data, the necessary routing is done. Therefore, the routing in the switch works statically based on the VL configuration table. There are other requirements that the ARINC 664 switching mechanism must implement. Since it is a deterministic network, a port-based maximum delay value is determined according to the needs of the network to be used. If the difference between the time the last bit of the packet arrives at the switch and the moment the last bit leaves the switch, those packets are dropped if the difference is above the determined maximum delay value. The packets are routed according to the destination port specified in the

VL configuration table if a delay is below the specified maximum delay. As another feature, if there is a connection error in any of the ports to which the packets will be forwarded, or if there is no connection, the packets going to the relevant port are dropped. As a final feature, the switch also includes a preemption mechanism. When defining VLs, their importance is also determined. ARINC 664 switches have two different priorities. These are high and low. In this priority mechanism, packets are put in different queues according to the priority value. They are taken from these queues in order of importance and forwarded to the output ports.

2.2.1.2.4 Configuration Function

ARINC 664 is a network which is designed offline. After the creation of topology, network traffic and specification is defined. A configuration table is created and uploaded to switch concerning network specification. Every VL has parameters it needs to run its switching, filtering, and policing mechanisms. The first of these is the range that the length of the incoming ARINC 664 packet should be. Therefore, L_{min} and L_{max} values are determined. These are essential parameters for filtering and policing. BAG and jitter values are the variables needed for the policing mechanism to work. Source, destination addresses, and Priority information are essential parameters for the switching mechanism [53]. Configuration tables with these specifications are loaded into the switch. Therefore, if frames suitable for these configurations are not sent from the end systems, the switch drops the sent packets.

There are also port-based configuration parameters in the switch. The purpose here is to determine the specifications that the switch should have. These can be exemplified as port speed, maximum delay time [40], port state.

2.2.1.2.5 Monitoring Function

The Monitoring module is used to display the critical situations in the switch, detect any fail status, and determine the modules that do not perform their duty in the switch. These data are transferred to MIBs (Management Information Base) determined by SNMP (Simple Network Management Protocol). As specified in the ARINC 664 standard [40], monitoring data should be updated at 100 ms intervals. Therefore, we need to keep MIBs up to date with a resolution of 100 ms with the SNMP protocol. We also have a fault/health mechanism in the monitoring module. This mechanism is responsible for detecting the vitally important errors identified earlier. In cases where these errors are detected, it is directly responsible for informing the relevant places with the error status. In other words, while it is sufficient to monitor some non-vital situations, it is necessary to report some situations to the relevant modules as soon as

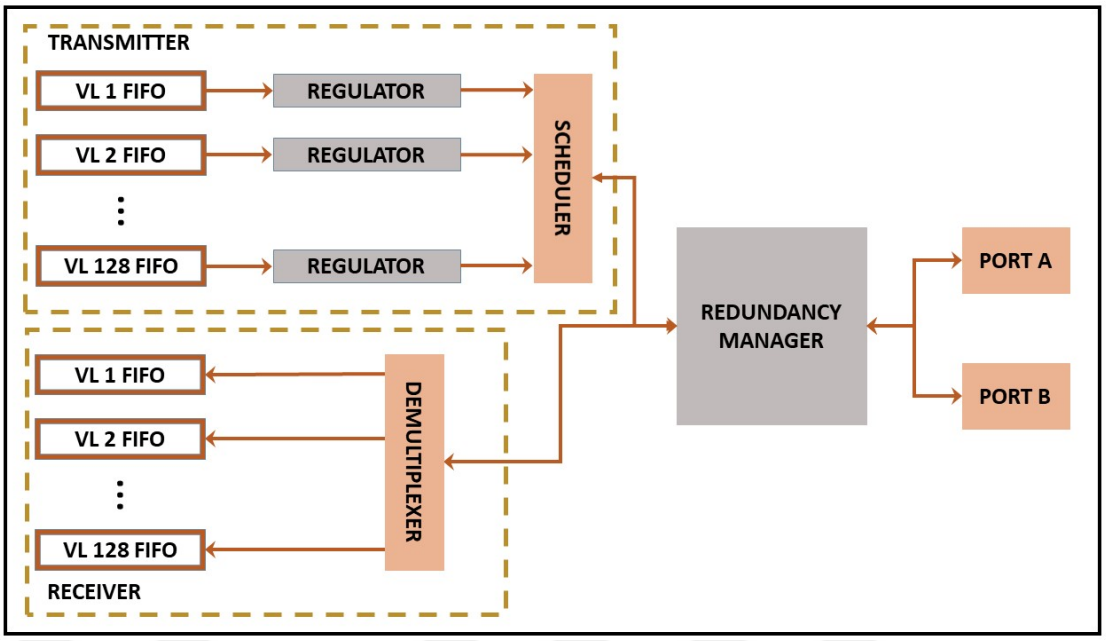


Figure 2.8. Internal structure of end system

an error occurs.

2.2.1.3 End System

The task of end systems is to send and receive data from various avionic elements such as application, sensors, and actuators according to the ARINC 664 protocol. As seen in Figure 2.8, the end system uses various modules while performing its tasks. The first of these modules is the transmitter, enclosed in a box and named as transmitter. Frames that need to be sent from the end system are prepared in the transmitter module and sent from the end system. The another module named as a receiver in the exact figure. Packets coming to the end system are forwarded to the upper layers by going through the processes mentioned in section 2.2.1.3.2. Finally, there is the redundancy module mentioned in section 2.2.1.3.3, which is used to ensure reliability.

2.2.1.3.1 End System Transmitter

The transmitter is shown in Figure 2.8, which prepares and sends the packets to be sent from the end system in accordance with the ARINC 664 protocol. First, the VL numbers of the frames to be sent are determined. Each VL number has its queues. Frames to be sent are transmitted to queues of their VL numbers. Having a different queue for each VL simplifies the task of the regulator module. Because of the regulator module, the packet sending interval is regulated according to the BAG and jitter values of the relevant VL number as shown in Figure 2.9. For example, VL 1 has a 1-millisecond

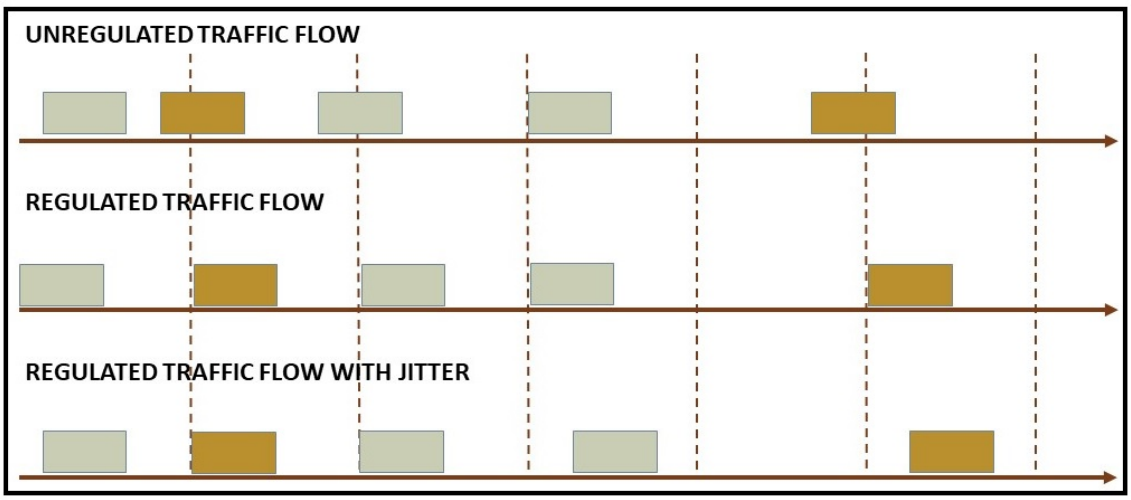


Figure 2.9. Traffic regulation by end system transmitter

BAG and no jitter. In this case, the required time difference between the two packets is one millisecond. These packets cannot be sent before one millisecond. However, if the queue is empty, there are no other packets, and the next packet arrives after one millisecond, the incoming packet can be sent as soon as it arrives. When the jitter value is 10 microseconds, there may be a maximum 10-microsecond deviation in the 1-millisecond BAG interval between the two packets. As a result, as seen in Figure 2.9, packets must be transmitted without disrupting the allocated BAG time interval when sending from the scheduler's output 2.9. However, there may be deviations up to the previously defined maximum jitter value. ARINC 664 frames go to the scheduler module after the regulator.

2.2.1.3.2 End System Receiver

Receiver is shown in Figure 2.8, which checks whether the packets coming to the end system are in accordance with the ARINC 664 protocol and sends these packets to the module that needs to be transmitted. The integrity checker controls packets with respect to VL, BAG, and redundancy requirements. After these checks, the frames are sent to the demultiplexer submodule. In the receiver module, as in the transmitter module, there are queues for each VL number. The demultiplexer sends the incoming packets to the corresponding queue of the VL number.

2.2.1.3.3 Redundancy Concept

The ARINC 664 protocol requires redundancy to ensure reliability. As seen in Figure 2.8, there is a redundancy module in front of the transmitter and receiver modules. From the transmitter perspective, the redundancy module creates a copy

of all the packets to be sent from the transmitter. Frames that are duplicates of each other must be understood in the destination end system. Therefore, it adds 1 byte of sequence number data to the end of identical packets. Sequence numbers take values sequentially from 0 to 255. When 255 is reached for the first time, they return to 1 and count up to 255 again. It returns to 1 and counts up to 255 until the system is reset or stopped and restarted. In cases where the system has just started to work or reset, the sequence number is started from 0. These two frames, which receive the sequence number data and are copies of each other, are sent simultaneously from the two Ethernet ports of the end system. These packets reach the same destination end system by passing through the switches in the topology, which are backups of each other. When the frames reach the destination end system, sequence numbers of frames are checked. If the sequence numbers are the same, the skewmax value are checked for redundant frames. The skewmax value is the difference when two frames with the same sequence number arrive at the end system. This difference is a configurable value. So if the frames come with a value greater than the configured value, both frames are dropped. Otherwise, both frames are considered valid. After passing all the checks, the redundant frame is dropped, and the original is transferred to the receiver module.

3
ORTHOGONAL ARRAYS

Orthogonal arrays (OAs) [11] are among mathematical tools to design an optimum mix of multiple variables in a set of experiments. They are effective in reducing the number of experiments, where, otherwise, a prohibitively large number of combinations would be needed[5–7][11]. They are classified as the same and mixed level OAs as described in Sections 3.1 and 3.2, respectively.

3.1 Same level OAs

In general, an OA is defined as a 4-tuple:

$$OA(N, F, L, S), \tag{3.1}$$

where

- N is the number of experiments (also referred to as *runs*) which represents the number of rows in an OA.
- F, called a *factor*, is the number of columns in an OA, where each column corresponds to a variable of an experiment.
- L is the number of *levels*, representing the number of distinct values that a factor can take. It is possible that, in a given experiment, different factors may possess different levels.
- S is called the *strength* of an OA ($0 < S \leq F$). An OA is designed in such a way that the number of rows needed to cover S-wise combinations of variables is given as L^S [5]. When $S = F$, all possible combinations of variables will be included in the experiment. A user can select the degree of combinations of variables to be considered by setting smaller values for S.

Table 3.1. Example for same level OA: an experiment with 3 factors each with 2 levels

Runs	f_1	f_2	f_3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

When all F factors have the same number of L levels, the OAs are referred to as the *same level OAs*. For a given experiment having F factors with L levels, there are L^F possible runs to cover all combinations of variables (assuming that all factors have the same number of levels L). OAs allow for reducing the number of runs to L^S by selecting an S value to cover S-wise combinations of variables. For example, all pairwise combinations of variables can be covered when S is set to 2. The number of runs can be further reduced as $N = \lambda \cdot L^S$ due to time limitations, where λ is a real number between 0 and 1 [9][54]. Selection of λ is determined by testers depending on the complexity and the number of parameters of a given system under test such that smaller values are preferred for complex systems.

Let us consider an experiment with $F = 3$ factors, each with $L = 2$ levels. Without using an OA, the number of runs required to cover all possible combinations of variables is 2^3 as shown in Table 3.1. We can observe that all possible situations between any two factors appear more than once in Table 3.1. For example, $(f_1 = 0, f_2 = 1)$ is listed twice for factors 1 and 2 in runs 3 and 4, respectively. Orthogonal arrays eliminate such repetitions and reduce the total number of runs for experiments with large number of factors and levels. If we design an OA for this example with all factors having 2 levels and selecting the strength as $S = 2$, we will have a total of $N = 2^2$ runs. In this case, the OA will be represented as $OA(4, 3, 2, 2)$ as shown in Table 3.2.

Table 3.2. Example for same level OA: runs for $OA(4,3,2,2)$

Runs	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

3.2 Mixed level OAs

In the OAs described in Section 3.1, all factors have the same number of levels. However, it is possible that, in real-life situations, some factors may have different number of levels to be tested. For example, in an ARINC 664 switch, mapping incoming traffic flows to outgoing ports may have a large number of possible assignments to be tested (e.g., up to 24 ports), whereas the number of different priorities for a flow may require much fewer options (e.g., two priorities of *High* and *Low*). OAs whose factors have different levels are called the *mixed level OAs*. The tuple defining a mixed level OA is given as [5]:

$$OA(N, L_1^{|F_1|} L_2^{|F_2|} \dots L_v^{|F_v|}, S) \tag{3.2}$$

where $|F_i|$ is the number of different factors with the same number of levels L_i for $(i = 1,2,\dots,v)$ and $\sum_{i=1}^{v} |F_i| = F$. N and S are defined in Eq. (3.1). For a factor F_i with the number of levels L_i, the number of possible runs will be ${L_i}^{|F_i|}$, which may be restricted by selecting an appropriate value of strength S as L_i^S as discussed above for the same level OAs. N, the number of runs, is selected in such a way that it is the least common multiplier of the number of S—wise factor level combinations. For a pairwise coverage, the combinations between any two factors can be generated as $L_i \cdot L_j$ for $i \neq j$ and $i, j = 1,\dots,F$. The generalization to S—wise combinations is studied in [5].

As an example, consider an experiment with $F = 5$ factors, where $F_1 = 4$ of them are with $L_1 = 2$ levels and one factor $F_2 = 1$ with $L_2 = 3$ levels. For the factors having 2 levels, the number of possible runs is $L_1^{|F_1|} = 2^4$, whereas, for the factor having 3 levels, there are $L_2^{|F_2|} = 3^1$ possibilities. Without using OAs, there are a total of $L_1^{|F_1|} \cdot L_2^{|F_2|} = 48$ runs for this example experiment. Suppose that we are interested in pair-wise combinations of these factors (i.e., $S = 2$). A mixed level OA can be defined as $(12, 2^4 3^1, 2)$, where $N = 12$ is calculated as the least common multiplier between the pair-wise combinations of any two factors. Since there are two levels (i.e., 2 and 3), the number of different combinations between any two factors can only be $2 \cdot 2 = 4$

or $2 \cdot 3 = 6$ with the least common multiplier is 12, which is the number of runs in Table 3.3.

Table 3.3. Example runs for a mixed level OA=$(12, 2^4 3^1, 2)$

Runs	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$	$\mathbf{f_4}$	$\mathbf{f_5}$
1	0	0	0	0	0
2	0	1	0	1	0
3	1	0	1	0	0
4	1	1	1	1	0
5	0	0	1	1	2
6	0	1	1	0	2
7	1	0	0	0	2
8	1	1	0	1	2
9	0	0	1	2	2
10	0	1	0	0	2
11	1	0	0	1	2
12	1	1	1	0	2

4
TESTING AVIONICS NETWORKS USING ORTHOGONAL ARRAYS

In this chapter, the test space and the use of the orthogonal array are discussed in general. In Section 4.1, the parameters that make up the test space of our equipment to be tested and the values it can take are mentioned. In Section 4.2, it is mentioned how much complexity is due to the values that the parameters that make up the test space can take. Finally, Section 4.3 describes the implementation of chained OA.

4.1 Test Space for DTN Switch

While traditional Ethernet switches perform switching function by routing frames from input ports to output ports by means of MAC tables, DTN switches utilize statically configured VL tables to route frames from input to output ports, where each frame carries VL identifier within its frame header to be used for switching purpose. An example DTN switch configuration, which is selected as the system under test (SUT), uses 4 ports and operates at the speed of 1 Gbps. The DTN switch includes filtering, policing, and switching features. A selected subset of parameters defining these features such as minimum and maximum frame lengths, BAG, and priority are shown in Table 4.1.

For this example switch configuration, there are eight different BAG values (in milliseconds) each indicating the time slot in which a VL needs to be transmitted. For example, VLs with BAG = 2ms are guaranteed to be transmitted within an interval of 2ms. L_{min} and L_{max} values define the minimum and maximum frame sizes, respectively ($L_{min} < L_{max}$). Although the ARINC 664 standard allows for frames between 64 and 1518 bytes [40], different minimum and maximum frame sizes within these limits are permitted. For example, if a VL selects $L_{min} = 100$ bytes, the switch will not permit a 99-byte packet to be transmitted in this VL. A small sample of L_{min} and L_{max} values are shown in Table 4.1. Determining the BAG values for all VLs, together with L_{min} and L_{max} values, will specify the maximum speed of operation for a switch.

In deterministic networks, it is important to classify packets by their importance so that they are prioritized in switch queues. For this example configuration, priority values are defined as *Low* and *High*. The number of incoming and outgoing VLs into a given port are selected to be eight distinct values between 192 and 3840.

4.2 Complexity of the Test Space

The number of VLs is an important parameter possibly indicating the traffic utilization rate of the ports and eventually the determinism features of a switch. Based on Table 4.1, there are $4 \cdot 8 \cdot 8 \cdot 8 \cdot 2 \cdot 8 \cdot 8 = 262{,}144$ different configurations for one VL. Since the maximum number of VLs in our example SUT implementation is 4096, the number of possible combinations of different VL configurations is prohibitively large. In an example configuration driven from Table 4.1, port 2 sends 1152 VLs to output ports 3 and 4 with BAG $= 64$, $L_{min} = 128$, $L_{max} = 1518$ and with high priority, whereas port 3 receives 384 VLs and port 4 the remaining 768 VLs (the total number of VLs sent and received will be equal among all ports). For the parameters shown in Table 4.1, the total number of combinations C for a given VL becomes

$$C = B^V \cdot L_s^V \cdot L_u^V \cdot p_i^V \cdot P \tag{4.1}$$

where B, V, L_s, L_u, p_i and P are the number of different values for BAG, VLs, L_{min}, L_{max}, priorities, and the number of ports, respectively. As an example with 1152 VLs for each port, the resulting complexity of the test space can be expressed using Eq. (4.1) as:

$$C = 8^{1152} \cdot 8^{1152} \cdot 8^{1152} \cdot 2^{1152} \cdot 4 \tag{4.2}$$

It is clear from Eq. (4.2) that it is not possible to generate a test suite with a feasible size without employing formal methods such as OAs. However, using OAs given in Table 4.1 may not generate test cases representing real-life usage of DTN switches. For example, a run generated from Table 4.1 with 2 ports, BAG $=$ 4ms, $L_{min} = 300$, $L_{max} = 1000$, and high priority implies that all VLs will have the same number of BAG, L_{min} L_{max} and priority values. But, in a realistic scenario, each VL can independently have any of the parameter values given in Table 4.1. In Section 4.3, we present an approach, where a sequence of OAs are employed to remedy this limitation.

Table 4.1. Sample configuration parameters for DTN switch

♯ of Ports	BAG (ms)	L_{min} (Bytes)	L_{max} (Bytes)	Priority	Incoming VLs	Outgoing VLs
1	1	64	1518	High	192	192
2	2	128	1400	Low	384	384
3	4	150	1300		576	576
4	8	200	1200		960	960
	16	300	1100		1152	1152
	32	400	1000		1920	1920
	64	512	900		2880	2880
	128	750	800		3840	3840

4.3 Using Chained OAs for DTN Test System

As described in Section 4.2, the DTN network features define a prohibitively large test space. Although OAs can be used to reduce the test space significantly by limiting the coverage to a subspace with pairwise combinations of factors, directly using the parameters in Table 4.1 may inadvertently eliminate important test cases representing real-life usage of DTN switches. In this thesis, we use *chained OAs*, where the test factors are divided into groups and OAs are generated sequentially such that the outputs of the first OA is used as the inputs for the next OA. Without loss of generality, let us first present the chained OA for two groups of input parameters, namely F_{g1} and F_{g2}, where $F_{g1} \cup F_{g2} = F$. With chained OAs, first, the parameters for F_{g1} that are considered to be more influential in system performance are selected and OA runs are generated with possibly higher values of strength S_{g1}. This way, suppression of different combinations is kept at minimum for this subset. After F_{g1} is handled, subsequent OAs are generated using the remaining parameters.

Let us consider an OA, called OA_{g1}, with N_{g1} runs, F_{g1} factors, each with level L_{g1}, and strength of S_{g1}:

$$OA_{g1}(N_{g1}, L_{g1}^{F_{g1}}, S_{g1}) \tag{4.3}$$

OA_{g1} generates an orthogonal array with N_{g1} rows and F_{g1} columns as follows:

$$\begin{matrix} n_{1,1} & n_{1,2} & \dots & n_{1,F_{g1}} \\ n_{2,1} & n_{2,2} & \dots & n_{2,F_{g1}} \\ \vdots & \vdots & \vdots & \vdots \\ n_{N_{g1},1} & n_{N_{g1},2} & \dots & n_{N_{g1},F_{g1}} \end{matrix}$$

where $n_{i,j}$ is the element of run i for factor j. Using the chained OAs approach, for each element $n_{i,j}$ of OA_{g1}, we generate a secondary OA, called $OA_{i,j}$:

$$OA_{i,j}(n_{i,j}, L_{g2}^{F_{g2}}, S_{g2})$$

where F_{g2}, L_{g2} and S_{g2} are the number of factors, levels and the strength selected for the secondary level orthogonal arrays, respectively ($S_{g2} \leq S_{g1}$). After applying the chained OAs, the total number of runs will be:

$$\sum_{i=1}^{N} \sum_{j=1}^{F_{g1}} (OA_{i,j}) \tag{4.4}$$

where $OA_{i,j}$ generates a matrix with dimensions of $(n_{i,j} \times F_{g2})$. For the general case with R subsets of F factors, where $F_{g1} \cup \cdots \cup F_{gR} = F$, subsequent OAs can be generated in a similar manner.

In testing a DTN switch, the number of VLs is the most critical parameter since it directly impacts the determinism of traffic flows through port utilization. Therefore, the first OA, called OA_{g1}, is defined to generate runs with different number of VLs for each of the four incoming and outgoing ports. Once these runs are generated by OA_{g1}, another OA, called OA_{g2}, is needed to cover the individual parameters of each of these VLs.

In defining OA_{g1}, the range of 1 to 4096 possible VLs for each of the four ports of a switch is arranged for this study by eight different values from 192 to 3840 as shown in Table 4.2: $OA_{g1}(256, 8^4, 2)$, where 256 is selected as the feasible number of runs and the strength is chosen as $S = 2$. Note that since the switch supports a maximum of 4096 VLs, any cases that exceed this upper limit in OA_{g1} are eliminated, resulting in 256 rows being reduced to 143.

VL configurations assigned by OA_{g1} to each switch port are determined by $OA_{i,j}$ using the five (5) parameters shown in Table 4.3. These parameters are selected as the most

Table 4.2. DTN switch variables for OAs: the number of incoming VLs to ports for OA_{g1}

Number of VLs			
Port 1	**Port 2**	**Port 3**	**Port 4**
192	192	192	192
384	384	384	384
576	576	576	576
960	960	960	960
1152	1152	1152	1152
1920	1920	1920	1920
2880	2880	2880	2880
3840	3840	3840	3840

Table 4.3 DTN switch variables for OAs: test parameters for VLs for $OA_{i,j}$

♯ of Ports	**BAG** (ms)	L_{min} **(Bytes)**	L_{max} **(Bytes)**	**Priority**
1	1	64	1518	High
2	2	128	1400	Low
3	4	150	1300	
4	8	200	1200	
	16	300	1100	
	32	400	1000	
	64	512	900	
	128	750	800	

influential ones on testing switch performance: $OA_{i,j}(n_{i,j}, 4^1 8^3 2^1, 2)$, where $n_{i,j}$ is the number of runs for element at i^{th} row and j^{th} column generated by OA_{g1}.

For the sake of simplicity, this study assumes that there are 4 ports in this implementation of a DTN switch and, hence, there are a total of $143 \cdot 4 = 572$ orthogonal arrays generated based on Table 4.2:

$$\sum_{i=1}^{143} \sum_{j=1}^{4} OA_{i,j} \tag{4.5}$$

where $OA_{i,j}$ is a matrix with the dimensions of $(n_{i,j} \times F_{g2})$, $n_{i,j}$ ranging from 192 to 3840 from Table 4.2 and $F_{g2} = 5$ is the number of columns in Table 4.3. Therefore, 572 different matrices with the dimensions of $n_{i,j} \times 5$ are generated for this simplified version of DTN switch, where each matrix corresponds to a different switch configuration. The total number of VL configurations ranges approximately from $143 \cdot 4 \cdot 192 = 109{,}824$ to $143 \cdot 4 \cdot 960 = 549{,}120$. Note that there are multiple test cases in each configuration as described in Section 5.

5
IMPLEMENTATION OF DTN TEST SYSTEM

In this chapter, the preparation process which are creating OAs and preparing AIM before the implementation of the test automation system and the implementation of the test automation system are explained step by step. First critical point is determining the parameters and obtaining orthogonal arrays. After that, the AIM test tool are used to test automation system with generated configuration file. These sections are described in Section 5.1.1 and 5.1.2. After the preparation process, the implementation of the test automation system is explained in the Section 5.2. Finally, the current state of the system is mentioned in Section 5.3.

5.1 Preparation to Implementation

There are two preparation phases before the implementation of the test system. One of them is to generate the OA configurations described in Section 5.1.1. Another is to adapt the AIM test tool described in Section 5.1.2 to the test environment to be applied.

5.1.1 Preparation of Configuration Files

In order to run the tests, it is necessary to prepare the inputs to be given to the system before the test environment is set up and the system starts to test. Therefore, important decisions affecting the implementation are taken during this process. The first of these decisions is to determine the parameters for the OA which is shown in Figure 5.2. While determining these parameters, it is one of the issues that the configuration files of the SUT to be tested and the AIM test equipment are the same. Because according to the configuration file, AIM is generated traffic and test the SUT. In addition, generating the configuration file in different formats for both AIM and SUT is an additional workload. As a result, a number of critical parameters which are common for AIM and SUT are decided upon.

```
run_size = 256
number_of_factors= 4
factor_levels= [8,8,8,8]
strength= 2

arrayclass=oapackage.arraydata_t(factor_levels, run_size, strength, number_of_factors)
print("arrayyclass:", arrayclass)
```

Figure 5.1. Parameters of oapackage python script

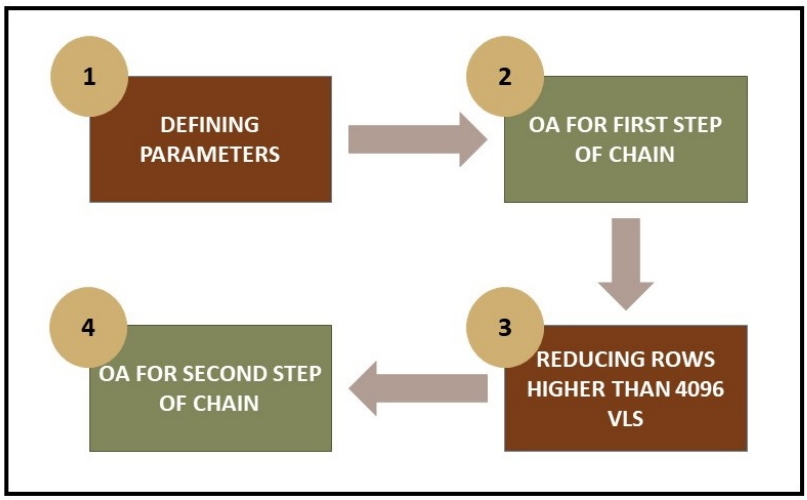


Figure 5.2. Architecture and information flow of DTN test system

Another important decision is how to focus on a scenario after the parameters are determined. The decisive factor for the scenario is that the equipment to be tested is the switch. Different traffic rates that will enter through the switch's ports have been a milestone that created the scenario. The parameter that determines the different traffic rate is the number of VL. Because the data rate is determined according to the number of VLs to be entered and the characteristics of these VLs. Therefore, the scenario required for the first step of the chained orthogonal array which is shown in Figure 5.2 as a second step, is formulated with Eq. 4.3 and performed with the parameters specified in the Table 4.2. This formulated OA was produced with the python library called *oapackage*. This python library accepts variables specified in Figure 5.1 as parameters. As stated in Eq. 4.3, a OA is obtained by the `oapackage.arraydata_t` command by providing parameters in the same way as the variables specified in Figure 5.1. The resulting OA is shown in Section 4.3 in a matrix format with 4 rows which are represented to ports and 256 rows which are represented to a numbers of VL configuration files.

Switch are designed to store the configuration of a maximum number of 4096 VLs. The sum of the four ports values in each row in the first OA result indicates the total number of VLs contained in a configuration file. The sum of some rows of OA is greater than 4096. Therefore, with a script written, rows which have a total VL number greater

than 4096 are removed from OA which is step three in Figure 5.2. In this way, 143 rows remain.

With the first orthogonal array, different number of VLs to enter from 4 ports are defined for 143 different configuration files. Thanks to step 2 of chained OA, configurations of VL numbers defined for each port will be obtained. Using the code in Figure 5.1, the new parameters are used to create the configuration files for VL in step 2 of the chained OA. The VL numbers determined for each port in the first OA are used as the `run_size` specified in the code in Figure 5.1 for the second OA, which is step four in Figure 5.2. Thus, the result of the first OA is used as the input of the second. The critical point here is that four separate OA's are generated for four existing ports in a row. The resulting four different OA files are combined end-to-end and become a single OA. If the VL number of the first file ends with 192, then the VL number of the two files is counted from 193 to 192 + `run_size`. Thus, 143 files which are input to the test system, are obtained.

5.1.2 Preparation of AIM Test Tool

AIM is a test equipment that contains various test scenarios to perform functional and performance tests. This test equipment consists of an ARINC 664 end system and pba.pro software. The software is working as a simulation tool, monitoring and analysis task. At the same time, it supplies the Run time informations, global statistics, VL-Activity and Event-Log information. AIM captures and monitors the frames coming from the switch chronologically [21].

AIM is important in facilitating the work done thanks to the contained test scenarios. It has classified all functional tests of the switch under ten headings. Four of these tests, which are the most basic of their classes, were used. These are filtering, policing, switching, and performance tests. Under each of these test titles, there are different test cases that test a feature of the switch related to that title [21]. The main purpose here is to run 143 configuration files that we produced for our own switch with AIM. The needed thing to do here is to create a script that will load these configurations into our switch and AIM.

5.2 System Architecture

A high level architecture of a DTN switch test system and its information flow are presented in Figure 5.3. First, OA model parameters representing realistic operation of the switch are determined as presented in Tables 4.1, 4.2 and 4.3. These parameters include the number of switch ports, possible number of VL combinations to be used

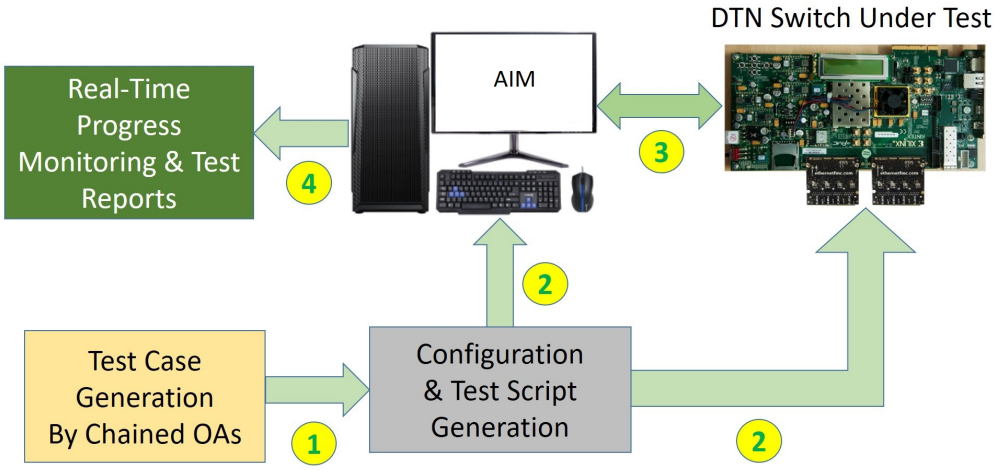


Figure 5.3. Architecture and information flow of DTN test system

for each port and the set of VL parameters to be included in OA array runs. The selection of switch parameters is an important step in test generation process since there are typically multiple trade-offs between a feasible number of test cases and a desired coverage of switch features that need to be decided together by the systems engineers, developers and test engineers. The current implementation of the DTN test system generates executable test scripts using the chained OAs method described in Section 4.3, which are applied to an SUT using a commercial tool called AIM [13].

Once the OA parameters are determined (e.g., the number of runs and the factors with their levels), the next step is to construct the OAs with the desired level of strength. In step 1 of Figure 5.3, the OA tables were generated using an open source Python library, called *oapackage* [12], whose capabilities were suitable for the study presented in this thesis. The chained OA method uses two different stages of OAs for this case study. In the first OA output, combinations of different number of VLs that can be assigned to each of the four switch ports are generated. The inputs used in this thesis for generating the first OA are given in Table 4.2. Then, each element of first OA output is used as the number of runs for the second stage OA to determine the remaining parameters such as BAG, L_{min}, L_{max}, and priority as shown in Table 4.3.

The outputs of the chained OA method are now ready to be converted to SUT configurations, which define the operational characteristics of a DTN switch. Since AIM is used as the tool to apply the tests to the SUT, similar conversions are performed to prepare the corresponding AIM configurations, as depicted in step 2 of Figure 5.3. With these configurations, test cases are executed by generating data frames, sending and receiving them to and from the SUT ports and checking their validity with respect to the expected outputs (step 3 in Figure 5.3).

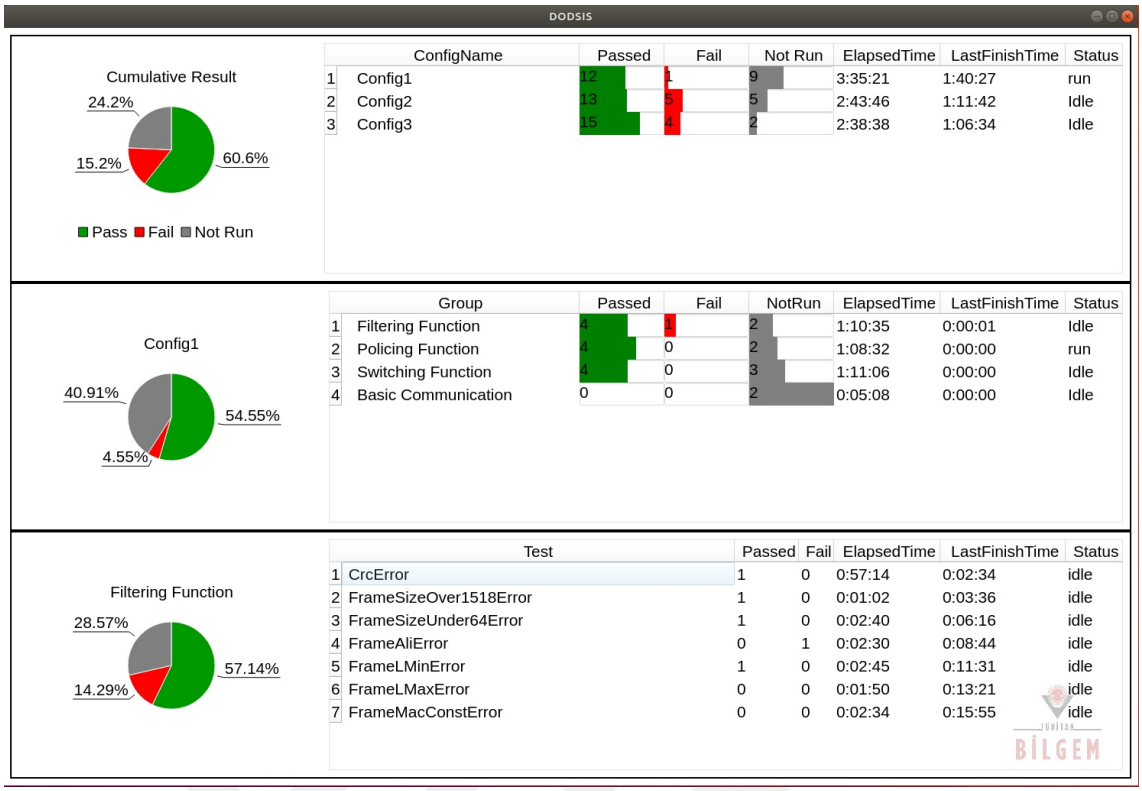


Figure 5.4. GUI sample for real-time status monitoring of DTN test system

Because there are large number of tests to be executed on SUT, logging the test results and generating appropriate reports are important for the test system (step 4 in Figure 5.3). In addition, since the tests are expected to run for many hours in a continuous manner, observing the status of the test progress in real-time is an important requirement for the test system. The GUI shown in Figure 5.4 is an example for test monitoring capabilities of the test system, which depicts switch configurations, the number of tests that are already run, pass/failure ratios, and the expected time of the completion. By clicking on any of the tabs on the GUI, test engineer will have access to more detailed outcomes of a given test.

5.3 Current Status of Test System

We estimate that, using chained OAs, the test suite will deploy hundreds of thousands of VLs to verify real-life scenarios that the SUT is expected to be exposed. Currently, the test suite includes 143 switch configurations, each of which represents a different setting for distinct usage of the DTN switch in the field. Each switch configuration will be run continuously for 3-4 hours, representing a realistic load for typical avionics applications.

6
RESULTS AND DISCUSSIONS

Test generation for network switches used in safety critical avionics systems is a significant challenge since these systems possess a large number of configuration options and high speed operation using thousands of virtual connections. There are prohibitively large number of test cases if one wishes to include all possible combinations of switch configuration parameters. One popular tool to reduce number of tests is OAs, especially suitable for complex industrial applications. In this thesis, we presented chained OAs method, where test parameters are divided into multiple groups and the OA generated for a group is used as an input for the next group. Independent of implementation or employed test architecture, chained OAs method can be applied to any SUT with large number of test parameters. For test generation of DTN switches, we defined two groups, where the first group deals with different number of virtual connections to be assigned to each switch port and the second one determines the remaining switch parameters such as transmission time gaps, minimum and maximum frame sizes and priorities. The test cases generated by the chained OAs method were converted into executable test scripts, which were applied to an SUT by a commercial test system.

Future extension of this work includes development of OAs that incorporate parameters defined for single and multiple end systems and reliability test suites for interoperability of switch and end systems for avionics applications. We expect that these extensions will be capable of representing real-life usage of deterministic avionics network systems and, hence, provide more effective test cases. We also plan to present a formal study to explicitly quantify the complexity and efficiency trade-off achieved by the chained OAs approach.

REFERENCES

[1] S. Schneele, F. Geyer, "Comparison of ieee avb and afdx," in *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, 2012, 7A1-1-7A1–9. DOI: 10.1109/DASC.2012.6382405.

[2] J. Athavale, R. Mariani, M. Paulitsch, "Flight safety certification implications for complex multi-core processor based avionics systems," Jul. 2019, pp. 38–39. DOI: 10.1109/IOLTS.2019.8854415.

[3] A. Arkusinski, "A method to increase the design assurance level of software by means of fmea," in *24th Digital Avionics Systems Conference*, vol. 2, 2005, 11 pp. Vol. 2-. DOI: 10.1109/DASC.2005.1563403.

[4] G. Igna, L. Dieudonne, S. Voss, B. Schatz, "Model-based deployment generation for safety-critical avionics systems," in *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2017, pp. 1–8. DOI: 10.1109/SIES.2017.7993394.

[5] A. S. Hedayat, S. N. J. A., J. Stufken, *Orthogonal Arrays: Theory and Applications*. Springer, ISBN 978-1-4612-1478-6, 1999, ISBN: 9781461214786.

[6] M. S. Phadke, *Quality Engineering Using Robust Design*. Prentice-Hall, ISBN 0-13-745167-9, 1989, ISBN: 0-13-745167-9.

[7] K. Gopalakrishnan, D. Stinson, D. Cheriton, "Applications of Orthogonal Arrays to Computer Science," in *Proc. of ICDM.*, 2006, pp. 149–164.

[8] G. Taguchi, S. Konishi, *Taguchi Methods: Orthogonal Arrays and Linear Graphs - Tools for Quality Engineering*. American Supplier Institute, Center for Taguchi Methods, ISBN 978-0-94-1243-01-8, 1987.

[9] R. Kacker, E. Lagergren, J. Filliben, "Taguchi's Orthogonal Arrays are Classical Designs of Experiments," *Journal of Research of the National Institute of Standards and Technology*, vol. 96, no. 5, p. 577, 1991. DOI: 10.6028/jres.096.034.

[10] E. D. Schoen, P. T. Eendebak, M. V. M. Nguyen, "Complete Enumeration of Pure-Level and Mixed-Level Orthogonal Arrays," *Journal of Combinatorial Designs*, vol. 18, no. 2, pp. 123–140, 2010, retrieved 15.05.2021. DOI: https://doi.org/10.1002/jcd.20236. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcd.20236. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcd.20236.

[11] S. Yazar, "Reconstruction of the Taguchi Orthogonal Arrays with the Support Vector Machines Method," *Balkan Journal of Electrical and Computer Engineering*, vol. 9, pp. 129–137, 2021, ISSN: 2147-284X. DOI: 10.17694/bajece.839449.

[12] P. Eendebak, *The Orthogonal Array Package*, retrieved 04.04.2021. [Online]. Available: http://www.pietereendebak.nl/oapackage/index.html.

[13] *AFDX / ARINC 664P7 Modules with a Difference*, retrieved 15.05.2021. [Online]. Available: https://www.aim-online.com/products-overview/modules/afdx-arinc664p7/.

[14] D. Chen, D. Song, "Afdx network performance testing," vol. 296, pp. 313–322, Jan. 2014. DOI: 10.1007/978-3-642-54236-7_35.

[15] F. Molina, P. Corral, M. Aljaro, G. Scals, A. Rodríguez, "Implementation of an afdx interface with zynq soc board in fpga," *Elektronika ir Elektrotechnika*, vol. 26, pp. 11–15, Oct. 2020. DOI: 10.5755/j01.eie.26.5.26008.

[16] D. Qu, B. Yang, T. Gao, L. Yuan, X. Chen, "Arinc 664 bus function test and its fault injection based on ethernet card," Jun. 2017, pp. 110–115. DOI: 10.1109/ICIEA.2017.8282824.

[17] W. Guangyun, Y. Bo, Y. Qinghua, "Key technologies of arinc664 bus testing," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2015, pp. 92–97. DOI: 10.1109/IMCCC.2015.27.

[18] R. Li, N. Huang, H. Liao, "Test profile design for avionics full-duplex switched ethernet," in *2014 Reliability and Maintainability Symposium*, 2014, pp. 1–6. DOI: 10.1109/RAMS.2014.6798501.

[19] R. L. Alena, J. P. Ossenfort, K. I. Laws, A. Goforth, F. Figueroa, "Communications for integrated modular avionics," in *2007 IEEE Aerospace Conference*, 2007, pp. 1–18. DOI: 10.1109/AERO.2007.352639.

[20] W. Shi-Kui, L. Wen, "Technical research and platform implementation of the protocol conformance test of afdx network switch," in *2018 International Conference on Networking and Network Applications (NaNA)*, 2018, pp. 227–231. DOI: 10.1109/NANA.2018.8648743.

[21] *AFDX Switch-Technical Solutions Testing White Paper*, retrieved 09.07.2012. [Online]. Available: https://www.aim-online.com/wp-content/uploads/2020/10/aim-ds-pba-pro-afdx-02-201009w.pdf.

[22] *Pba.pro-AFDX® Analysis Software*, retrieved 09.10.2020. [Online]. Available: https://www.aim-online.com/wp-content/uploads/2019/06/aim-afdx-switch-testing-whitepaper-reva-u.pdf.

[23] I. Moir, A. Seabridge, M. Jukes, *Civil Avionics Systems*, ser. Aerospace Series. Wiley, 2013, ISBN: 9781118536728. [Online]. Available: https://books.google.com.tr/books?id=8XFwAAAAQBAJ.

[24] R. Ramaker, W. Krug, W. Phebus, "Application of a civil integrated modular architecture to military transport aircraft," in *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, 2007, 2.A.4-1-2.A.4–10. DOI: 10.1109/DASC.2007.4391845.

[25] C. B. Watkins, R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," in *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, 2007, 2.A.1-1-2.A.1–10. DOI: 10.1109/DASC.2007.4391842.

[26] A. Uncu, S. Üzümcü, A. A. Mert, "Cyber security concerns regarding federated, partly ima and full ima implementations," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–5. DOI: 10.1109/DASC43569.2019.9081614.

[27] E. A. Papon, N. Verma, "Aviation data transmission-moving from electrical to optical domain: Review of arinc 429 and 629 standards," Dec. 2012.

[28] J. Perez-Mato, R. Perez-Jimenez, J. Tristancho, "Optical wireless interface for the arinc 429 avionics bus: Design and implementation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 6, pp. 15–21, 2013. DOI: 10.1109/MAES.2013.6533740.

[29] R. Vyas, A. Bhattacharjee, A. Khandelwal, "Development of soc for interfacing avionics units with arinc 429 bus," in *2016 International Conference on Communication and Electronics Systems (ICCES)*, 2016, pp. 1–5. DOI: 10.1109/CESYS.2016.7889931.

[30] Y. Luo, X. Zhao, "Design and research of an arinc 429 bus testing system," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, 2014, pp. 194–198. DOI: 10.1109/CCDC.2014.6852143.

[31] L. Cohen, "Testing the subscriber interface to the arinc 629 current mode bus," in *Proceedings of AUTOTESTCON '94*, 1994, pp. 653–661. DOI: 10.1109/AUTEST.1994.381555.

[32] A. Gabillon, L. Gallon, "Availability constraints for avionic data buses," in *First International Conference on Availability, Reliability and Security (ARES'06)*, 2006, 7 pp.–131. DOI: 10.1109/ARES.2006.35.

[33] P. Topolsky, P. Dycka, M. Smatana, P. Bojda, J. Bajer, "Mil-std-1553 message transmission and reception using fpga," in *2019 International Conference on Military Technologies (ICMT)*, 2019, pp. 1–5. DOI: 10.1109/MILTECHS.2019.8870014.

[34] D. Duvvuri, P. V. Sankara Rao, J. Chattopadhyay, "100-mbps transceiver for enhanced mil-std-1553," in *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2014, pp. 316–319. DOI: 10.1109/APCCAS.2014.7032783.

[35] C. Hou, S. Wang, Q. Wang, H. Zhang, "Performance analysis of high-speed mil-std-1553 bus system using dmt technology," in *2013 8th International Conference on Computer Science Education*, 2013, pp. 533–536. DOI: 10.1109/ICCSE.2013.6553968.

[36] J. Windsor, M.-H. Deredempt, R. De-Ferluc, "Integrated modular avionics for spacecraft — user requirements, architecture and role definition," in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, 2011, 8A6-1-8A6–16. DOI: 10.1109/DASC.2011.6096141.

[37] B. Gul, O. F. Gemici, İ. Hokelek, M. Garip, "Latency analysis of switchless arinc 664 using network calculus," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, 2021, pp. 1–4. DOI: 10.1109/SIU53274.2021.9477961.

[38] T. V. Mani, S. Maitra, P. BhanuSrinivas, K. L. R. Sekhar, S. V. Lakshmi, R. M. Guptha, "Integrated modular avionics for missile applications," in *2019 International Conference on Range Technology (ICORT)*, 2019, pp. 1–5. DOI: 10.1109/ICORT46471.2019.9069629.

[39] J.-P. Moreaux, *Data transmission system for aircraft*, U.S. Patent US6925088B1, Nov. 1999.

[40] *ARINC 664, P7: Avionics Full Duplex Switched Ethernet (AFDX) Network*. INC Aronautical Radio, 2005.

[41] E. C. Akpolat et al., "Genetic Algorithm Based ARINC 664 Mixed Criticality Optimization Using Network Calculus," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6. DOI: 10.1109/ICCWorkshops50388.2021.9473569.

[42] M. N. Durceylan, O. F. Gemici, I. Hokelek, H. Asmer, "Reconfigurable arinc 664 end system implementation on fpga," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, 2020, pp. 1–4. DOI: 10.1109/SIU49456.2020.9302483.

[43] S. Kaya *et al.*, "Delay optimization for switchless arinc 664 mesh networks with cyclic dependencies," in *2021 IEEE Latin-American Conference on Communications (LATINCOM)*, 2021, pp. 1–6. DOI: 10.1109/LATINCOM53176.2021.9647752.

[44] M. Yeniaydin et al., "Priority Re-assignment for Improving Schedulability and Mixed-Criticality of ARINC 664," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6. DOI: 10.23919/IFIPNetworking52078.2021.9472784.

[45] A. Bouillard, M. Boyer, E. Le Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. John Wiley & Sons, ISBN 978-1-119-56341-9, 2018, ISBN: 978-1-119-56341-9.

[46] X. Tang, Z. Luo, G. Qu, "Worst-case end-to-end delay analysis for an afdx evolution network," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 78–82. DOI: 10.1109/ICCC51575.2020.9344991.

[47] X. Gao, Y. Fu, T. Liu, W. Jiafang, "A new re-shaping method for reducing delay jitter on afdx network," in *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce*, 2014, pp. 29–33. DOI: 10.1109/ICITEC.2014.7105565.

[48] M. Adnan, J.-L. Scharbarg, C. Fraboul, "Minimizing the search space for computing exact worst-case delays of afdx periodic flows," in *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, 2011, pp. 294–301. DOI: 10.1109/SIES.2011.5953673.

[49] K. Bisson, T. Troshynski, "Switched ethernet testing for avionics applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 5, pp. 31–35, 2004. DOI: 10.1109/MAES.2004.1301224.

[50] W. Shi-Kui, L. Wen, "Technical research and platform implementation of the protocol conformance test of afdx network switch," in *2018 International Conference on Networking and Network Applications (NaNA)*, 2018, pp. 227–231. DOI: 10.1109/NANA.2018.8648743.

[51] N. Warden, "Overview and effect of deterministic ethernet on test strategies," in *2017 IEEE AUTOTESTCON*, 2017, pp. 1–3. DOI: 10.1109/AUTEST.2017.8080468.

[52] J. Zhang, S. Qiao, D. Li, G. Shi, "Modeling and simulation of edf scheduling algorithm on afdx switch," in *2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2011, pp. 1–4. DOI: 10.1109/ICSPCC.2011.6061640.

[53] X. Li, N. Huang, F. Zhao, "A genetic algorithm based configuration optimization method for afdx," in *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)*, 2014, pp. 440–444. DOI: 10.1109/ICRMS.2014.7107235.

[54] H. Bayrak, A. Alhan, "On the Construction of 2-Symbol Orthogonal Arrays," *Hacettepe Journal of Mathematics and Statistics Volume*, vol. 31, pp. 57–62, Jan. 2002.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. M. Altuntas, M. Seker, P. Kısla, E. Can, M. Demir, I. Hokelek, E. Akdogan, “Testing Deterministic Avionics Networks Using Orthogonal Arrays,” in UBICOMM 2021,Oct. 2021, pp. 1–6, ISBN: 978-1-61208-886-0.

Awards

1. IARIA UBICOMM 2021, Best Paper Award