OPTIMIZING MESSAGE TO VIRTUAL LINK ASSIGNMENT IN AVIONICS FULL-DUPLEX SWITCHED ETHERNET NETWORKS

By

Joseph Klonowski

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Master of Science

2019

ABSTRACT

OPTIMIZING MESSAGE TO VIRTUAL LINK ASSIGNMENT IN AVIONICS FULL-DUPLEX SWITCHED ETHERNET NETWORKS

By

Joseph Klonowski

Avionics Full-Duplex Switched Ethernet (AFDX) is an Ethernet-based data network that provides deterministic performance, high reliability, and lower costs and development time by utilizing commercial off-the-shelf networking components. As AFDX networks have become the common solution for network design in aviation, analyses for optimizing different aspects of the network are continually being evaluated. There are two main types of solutions to improving network performance: changes to the physical layer and changes to the logical layer. Because the physical network is setup prior to defining the data that is transferred on the network, logical layer optimization becomes important and is often the only viable solution. Previous research has explored optimization of different aspects of the logical solution for a given target (whether it be latency or bandwidth), however, an approach for a customizable target using optimization techniques has not been attempted. In this work, we provide an overview of AFDX networks and discuss factors engineers consider while optimizing the network. Previously researched solutions are evaluated for effectiveness. We identify the need for an optimization solution that allows for a customizable objective to account for both message latency and bandwidth. To fill this gap, we consider the problem of assigning messages to *virtual links*, which are configurable, logical unidirectional links from publishing end systems to one or more subscribing end systems. We propose a flexible framework based on particle swarm optimization (PSO) that performs message to virtual link assignment in AFDX networks to optimize a user-defined objective. We discuss and provide results on PSO optimization for a range of hyperparameters. Finally, results for a sample swarm are presented to prove the feasibility and usefulness of the proposed approach.

ACKNOWLEDGMENTS

I would like to express my thanks and appreciation to my advisor Dr. Nihar Mahapatra. He is the person who encouraged me to pick up this research in the beginning. This research would not have been possible without his support. I would like to thank Dr. Fathi Salem and Dr. Lalita Udpa for serving on my committee. I would also like to thank my wife, family, friends, and colleagues for the constant advice and encouragement throughout my research. Finally, I would like to thank GE Aviation Systems LLC for providing the resources in order to conduct this research.

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

KEY TO ABBREVIATIONS x

CHAPTER 1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Problem Background 2
- 1.3 Researched Solutions 3
- 1.4 Contributions 3
- 1.5 Thesis Organization 3

CHAPTER 2 AVIONICS NETWORKING BACKGROUND 5
- 2.1 Avionics Full-Duplex Switched Ethernet 5
- 2.2 Virtual Link 6
  - 2.2.1 Virtual Link Grouping 9
  - 2.2.2 Message Grouping in Realistic Scenario 11
  - 2.2.3 Virtual Link Frame Structure 12
  - 2.2.4 Bandwidth, Throughput, and Latency 13
  - 2.2.5 Virtual Link Constraints 14
- 2.3 Remote Data Concentrator and Remote Gateway 16
- 2.4 Interface Control Document 17

CHAPTER 3 CURRENT DESIGN APPROACH AND RELATED WORK 18
- 3.1 Optimal BAG and MTU Values 19
- 3.2 Message Grouping and Virtual Link Aggregation 21
- 3.3 Optimized Transmission Scheduling 22
- 3.4 New Contribution For Research 24

CHAPTER 4 PARTICLE SWARM IMPLEMENTATION 25
- 4.1 Particle Swarm Optimization Introduction 25
- 4.2 Particle Swam Optimization Mathematics 26
- 4.3 Modeling Message to Virtual Link Assignment 28
- 4.4 Framework for Implementation 30
- 4.5 Scope of Algorithm 32

CHAPTER 5 NETWORK IMPLEMENTATION AND RESULTS 34
- 5.1 Network Used for Testing 34
- 5.2 Hyperparameter Selection 35
- 5.3 Optimization Results 42

CHAPTER 6 CONCLUSION AND FUTURE WORK 43
BIBLIOGRAPHY 45

LIST OF TABLES

Table 2.1: Example Network Data 5

Table 2.2: Possible Groupings into Virtual Links 10

Table 4.1: Example Particle 28

Table 4.2: Example Particle With Limited Virtual Links 29

Table 5.1: Test Patterns 36

LIST OF FIGURES

Figure 2.1: Simple AFDX Network 6

Figure 2.2: BAG and MTU 7

Figure 2.3: sub-VL Transmission 7

Figure 2.4: Example VL Transmission 8

Figure 2.5: Jitter Example 9

Figure 2.6: VL Grouping Example 10

Figure 2.7: Virtual Link Frame 12

Figure 2.8: Simple Network Example 16

Figure 3.1: Example Aircraft AFDX Network 18

Figure 3.2: Branch and Bound Results [2] 20

Figure 3.3: Message Grouping Results [4] 22

Figure 3.4: Scheduling Algorithm Results [3] 24

Figure 4.1: Particle Swarm Optimization Example 26

Figure 4.2: Optimization Block Diagram 30

Figure 4.3: Example Grouping in XML 31

Figure 4.4: Scope of Algorithm 33

Figure 5.1: Test Network Design 34

Figure 5.2: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .1, .9\}$ 37

Figure 5.3: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .3, .7\}$ 38

Figure 5.4: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .7, .7\}$ 39

Figure 5.5: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .7, .3\}$ 40

Figure 5.6: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .9, .1\}$ 41

Figure 5.7: 10 Particles 100 Iterations $\{m, c_1, c_2\} = \{0, .7, .7\}$ 42

KEY TO ABBREVIATIONS

AFDX Avionics Full-Duplex Switched Ethernet

B Bandwidth

BAG Bandwidth Allocation Gap

DAL Design Assurance Level

ES End System

FIFO First In First Out

I/O Input / Output Data

ICD Interface Control Document

IP Internet Protocol

L Latency

MTU Maximum Transmission Unit

RDC Remote Data Concentrator

RGW Remote Gateway

RSN Received Sequence Number

TP Throughput

UDP User Datagram Protocol

VL Virtual Link

CHAPTER 1

INTRODUCTION

Avionics Full-Duplex Switched Ethernet (AFDX) is an Ethernet-based data network that provides deterministic performance, high reliability, and lower costs and development time by utilizing commercial off-the-shelf networking components. It was designed specifically to meet the stringent integrity and availability requirements required in aviation [16]. This widely accepted solution implements the ARINC 664-Part 7 [1] standard required in the avionics networking industry. The interest in AFDX networks as the network solution is due to its high speed, low cost, high flexibility, and reduced weight because of less wiring [20]. As AFDX networks have become the common solution for network design in aviation, analyses for optimizing different aspects of the network are continually being evaluated. With growth in the amount of data on the network, network performance optimization has gained increasing significance. There are two main types of solutions to improving network performance: changes to the physical layer and changes to the logical layer. Physical layer changes might yield better results for increasing the capacity of a network, however, they are much more difficult to implement once the aircraft's design has begun. Because the physical network is setup prior to defining the data that is transferred on the network, logical layer optimization becomes important and is often the only viable solution.

1.1 Motivation

In today's world, people want access to more data and they want to receive that data quicker. The aviation industry is no different. With the introduction of more complex aircraft, systems are publishing more data than ever. A recent relatable example of this is the introduction of entertainment systems on airplanes. The entertainment data is published on the AFDX network and is sharing the same physical network as more important data such as flight control data. So the introduction of this new data is decreasing the amount

of available resources of the AFDX network. Given that aircraft are not using the most up-to-date hardware (because aircraft development is much slower than commercial networking development), it becomes a problem of how can we fit more data on legacy hardware.

Aviation is an industry where space is limited, weight is heavily scrutinized, and power is of utmost importance, any area where savings can be found is extremely important. To get an idea of the importance, one extra pound on an airplane has the cost of roughly $1M USD over the life-cycle of the fleet of an aircraft [12] [13]. For example, with the Airbus A380 containing 500 km of cables, the pressure to more effectively use a given physical design of a network continues to be scrutinized [11]. However many times due to time and budget constraints, optimization is not considered once development is complete. This is where developing a smarter logical layer solution for a given physical layer solution can help solve this optimization problem. An optimal logical layer solution will allow for more data to be transmitted on the AFDX network using legacy networking hardware. The importance of this is that it makes it easier to route data through the network without contention and doesn't violate the ARINC 664-Part 7 availability and integrity requirements [1].

1.2 Problem Background

The reasoning for finding a new network implementation prior to AFDX's introduction was because legacy network solutions (such as ARINC 429) had reached their limits for performance and design complexity [21], [24]. The amount of data on AFDX networks is continuously growing due to a continuous increase in the number of electronic components publishing and subscribing to data on the AFDX network [10]. This increase in data on the network brings about the problem of how the extra data will affect network performance. With the constant increase in the amount of data on the network, the physical network is approaching its limits for performance. An optimal logical network will also allow more data to be put onto the current hardware used in aviation (keeping in mind that this is legacy hardware since aircraft development is not as fast as commercial network development).

There currently isn't a great understanding of how different logical choices impact latency and throughput of an aircraft's network on a large scale. With the ever-growing amount of data on airplanes (both safety critical and non-safety critical), ensuring that an optimal solution is designed is more important than ever. Given the competitive nature of the aviation industry, finding the optimal network solution will be revolutionary to a future, and in some cases current, problem in the avionics field.

1.3 Researched Solutions

Optimizing the logical layer solution has a vast majority of possibilities. Three options that this research will focus on are virtual link construction, message to virtual link assignment, and virtual link scheduling. Previously research methods for optimizing these options are summarized with an overview of the results of studies on different algorithms. The gap in research for a customizable objective function will be shown.

1.4 Contributions

This thesis will give some insight into network development and implementation in the aviation industry. An exhaustive search of research for optimizing AFDX networks was done and concluded that optimization for a user-defined objective function, accounting for both message latency and bandwidth, has not been attempted. To fill this gap, we consider the problem of assigning messages to virtual links. We propose a flexible framework based on particle swarm optimization (PSO) that performs message to virtual link assignment in AFDX networks to optimize a user-defined objective. We discuss and provide results on PSO optimization for a range of hyperparameters.

1.5 Thesis Organization

This thesis is organized as follows. In Chapter 2, background and definitions of Avionics Full-Duplex Switched Ethernet networks are provided. These definitions are important to understand the research that is being done. In Chapter 3, previously researched solutions

for optimization of AFDX networks will be outlined. Each solution will be introduced and explained, then the results of the research will be shown and areas for improvement will also be discussed. In Chapter 4, an introduction to Particle Swarm Optimization will be presented. An explanation of how Particle Swarm Optimization will be used to solve the optimization problem of message to virtual link assignment will be given. An overview of the framework that was created will be covered along with how it can be translated for future research. In Chapter 5, the network that will be used as the test setup will be explained. Results of the algorithm will be shown as well as limitations with the research. Finally in Chapter 6, the conclusion of the research and future work will be discussed.

CHAPTER 2

AVIONICS NETWORKING BACKGROUND

In order to fully understand the problem, it is important to understand common terminology used throughout this research.

2.1 Avionics Full-Duplex Switched Ethernet

Avionics Full-Duplex Switched Ethernet (AFDX) is the implementation of the ARINC 664-Part 7 standard for aircraft networks [1]. This is the aviation equivalent of a standard Ethernet network with more stringent bandwidth and redundancy requirements. Within the AFDX network, *end systems* (any publisher and/or subscriber of data), switches, and physical links make up the physical layer while virtual links outline the logical layer. A simple example of an AFDX network can be seen below in Figure 2.1. An example of the type of data that is transmitted within the AFDX network can be found in Table 2.1. Note that *Design Assurance Level* (DAL) is a label determined from a safety assessment process and hazard analysis by examining the effects of a failure condition in the system [15]. This label correlates with the criticality of messages. The most stringent DAL is DAL A, while the least stringent DAL is DAL E.

Example AFDX Data	DAL
Flight Control Systems	A
Braking Systems	B
Backup Landing Systems	C
Ground Navigation Systems	D
Entertainment Systems	E

Table 2.1: Example Network Data

2.2 Virtual Link

A *Virtual Link* (VL) is a configurable and logical unidirectional link from one publishing end system to one or more subscribing end systems. Virtual links provide the avenues for messages to go from end system to end system. An easy way to think of a virtual link is to think of it as a path through the physical network from a publisher to subscriber. See Figure 2.1 for an example of how virtual links fit into a simple AFDX network. The figure includes four end systems. ES1 and ES2 are publishing data on the AFDX network, while ES3 and ES4 are subscribing to the data on the AFDX network. Note that end systems can subscribe to data on the AFDX network, publish data on the AFDX network, or both.

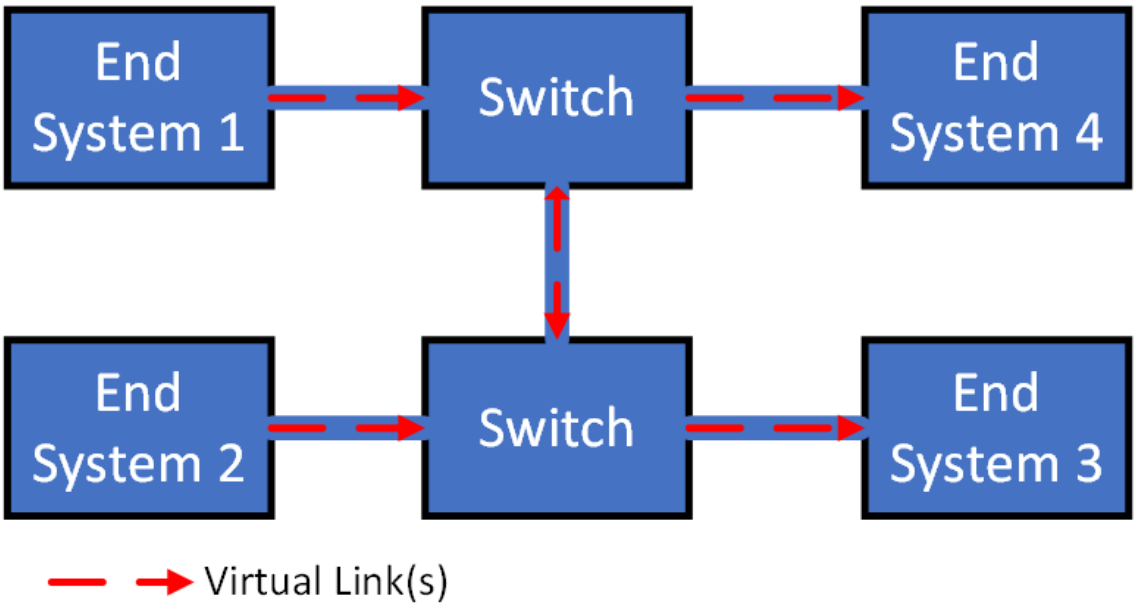

Figure 2.1: Simple AFDX Network

Each VL has certain characteristics which define its behavior. First is the *Bandwidth Allocation Gap (BAG)*. BAG is the minimum time interval between successive frames of a virtual link. A given BAG, assigned to each VL, can range anywhere from 1ms to 128ms in power of two increments. *Maximum Transmission Unit* (MTU) is the size of the message (bytes) in each VL's frame (size of the VL). BAG and MTU are used in combination to determine the scheduling of VLs for transmission to the AFDX network. This can be seen

graphically in Figure 2.2.

Figure 2.2: BAG and MTU

A given VL can be further partitioned into up to a maximum of four sub-VLs. The sub-VLs are aggregated into a FIFO queue prior to transmission to the AFDX network. The goal of sub-VLs is to optimize the bandwidth utilization of a given VL. This can be seen in Figure 2.3, which has three sub-VLs. They're put into the FIFO queue and transmitted out to the AFDX network. This is more effectively using the VL's bandwidth utilization because of the ability to send out separate data individually, rather than sending larger data frames all at once.

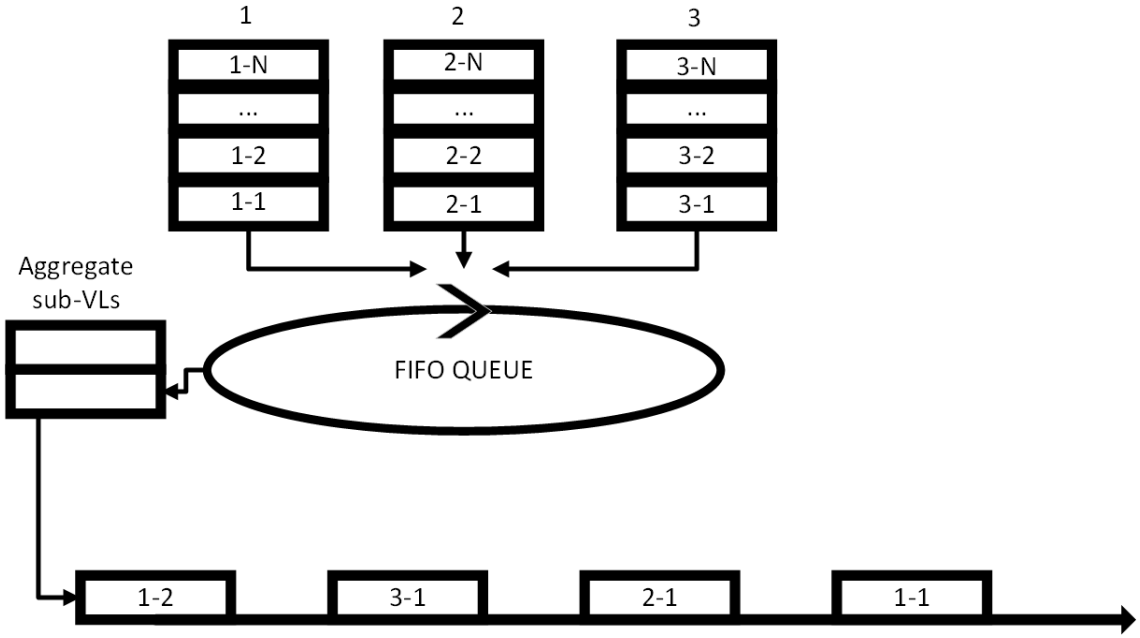

Figure 2.3: sub-VL Transmission

The transmission of VLs can get complex when an ES has multiple VLs which need to be sent out to the AFDX network. This is where two common techniques can help with scheduling, namely First-In-First-Out (FIFO) queues and a generic scheduling technique. FIFO queues are simplistic in that the first VL that is ready to be transmitted by an ES will be transmitted first (while still not violating the BAG requirement discussed above). Subsequent VLs will be transmitted as they are received, while still adhering to the BAG requirement.

Generic scheduling techniques are more in-depth. This scheduling involves assigning each VL a time-slice in which it is able to transmit messages to the AFDX network. If a VL is ready to be transmitted, it will be sent. However if it is not ready to be transmitted, a ripple effect could be propagated through the network. The goal of creating this schedule is a more consistent scheduling mechanism with more effective use of the physical link's bandwidth.

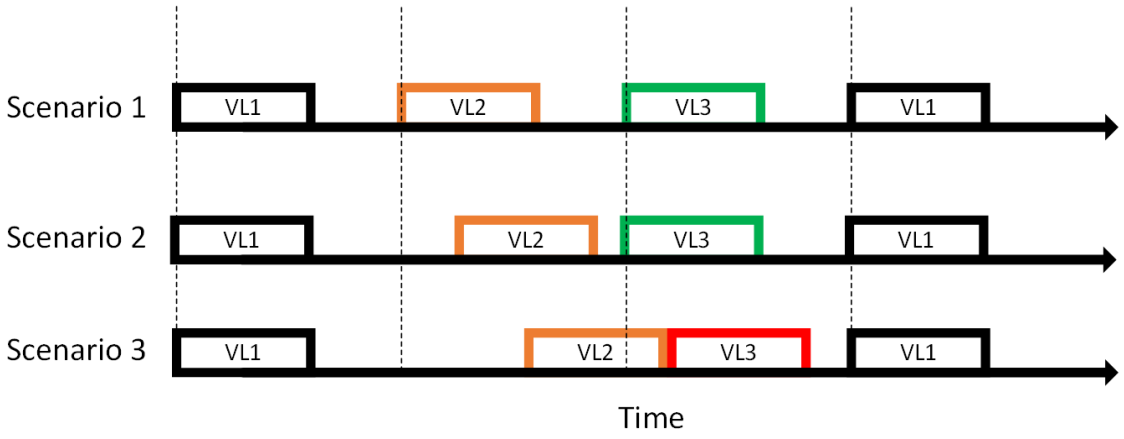

Figure 2.4: Example VL Transmission

The example in Figure 2.4 shows the transmission behavior for a generic scheduling methodology. In each of the three scenarios, VL1 and VL3 are ready to transmit at their time slice. However VL2 isn't ready to transmit until a later time in each scenario. In the first scenario, each VL is ready to transmit at its appropriate time slice. In the second scenario, the delay in VL2 doesn't affect the normal transmission time of VL3, so VL3 is still transmitted at the same time. However in the third scenario, VL2's delay runs over the

regular transmission time of VL3. This causes a ripple effect where VL3 is now transmitting a bit delayed from the previous two scenarios. In the event of a relatively lightly loaded network, such as in these scenarios, the ramifications of the ripple effect are not too serious. In the event of a heavily loaded network where each time slice is allotted to a particular VL with little buffer in between, the ripple effect can cause large amounts of delay. This delay leads into the concept of jitter.

The last characteristic of VLs needed to understand this research is jitter. *Jitter* is the difference between the minimum and maximum time from when a publisher sends a VL to when the subscriber receives the VL. This means that it is the variation of the latency of a particular VL. Jitter is caused by contention with other VLs on the AFDX network. Because AFDX networks are a real-time network, jitter is heavily scrutinized and has constraints levied upon it defined by ARINC 664-Part 7. A graphical view of jitter can be seen in Figure 2.5.

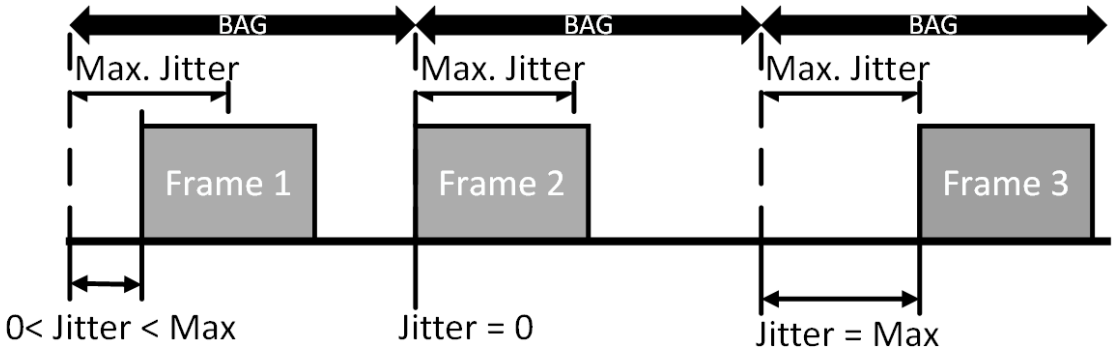

Figure 2.5: Jitter Example

2.2.1 Virtual Link Grouping

Virtual links define the path from a publisher to subscribers, however what if there is more than one VL sharing a similar path from a publisher to similar subscribers? This is where the concept of grouping similar messages is important.

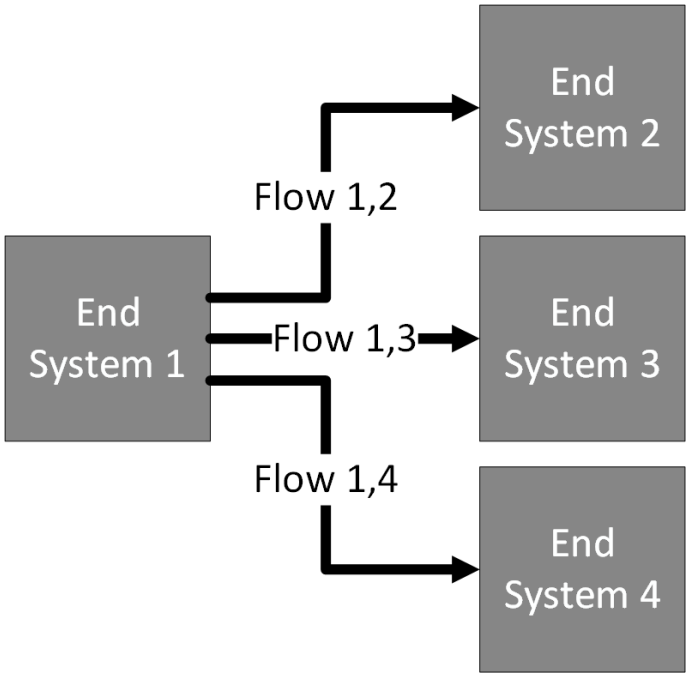

Figure 2.6: VL Grouping Example

As an example, imagine the scenario in Figure 2.6. The Flight Management System (End System 1) is sending a message to the Flight Data Recorder (End System 2), Landing Gear Controller (End System 3), and Flaps Controller (End System 4). Looking at the example in Figure 2.6, there are six possible ways to group these messages (flows). They can be found in the table below.

VL1	VL2	VL3
$F_{1,2}, F_{1,3}, F_{1,4}$	empty	empty
$F_{1,2}, F_{1,3}$	$F_{1,4}$	empty
$F_{1,2}, F_{1,4}$	$F_{1,3}$	empty
$F_{1,3}, F_{1,4}$	$F_{1,2}$	empty
$F_{1,2}$	$F_{1,3}$	$F_{1,4}$

Table 2.2: Possible Groupings into Virtual Links

Grouping of virtual links makes scheduling of transmissions easier for end systems. The main drawback is that the size of the virtual link increases with each grouped virtual link. For example, the first row of the table above groups all of the messages into one VL. That

VL will then be sent to all three end systems. The subscribing end systems will still receive the data intended for the it, but it is reserving more bandwidth on the physical link than is required. Grouping creates an inefficiency because the physical links end up reserving more bandwidth than is necessary. Grouping messages into VLs makes scheduling easier by the publishing end system, which will reduce jitter. The cost is that the overall bandwidth of the system will increase. The other extreme would be to have each message in its own VL. Then each end system will get only the data that it needs, however scheduling is tougher for the publishing end system. When the number of messages increases, scheduling will be more difficult. Also there are sometimes requirements levied upon the publishing end system capping the number of VLs transmitted. If the number of messages needed to be published exceeds the maximum allotted VLs for the end system, this extreme case of one message per VL isn't feasible.

Once messages have been grouped together into VLs, the VLs can then be partitioned into sub-VLs, as shown in Figure 2.3. These two design decisions affect the VL routing, scheduling, throughput, and latency for the network.

2.2.2 Message Grouping in Realistic Scenario

In looking at more realistic scenarios, the number of messages published by a remote gateway will be more in the hundreds to thousands. This means that there are potentially thousands of messages that have the possibility of being grouped together in VLs. Because the number of messages grouped together in VLs is a design decision, the number of VLs that exist throughout the network varies based upon design decisions. This leads to the question, how many grouping pairs are possible? The solution to this is not trivial, since the number of VLs is variable, as well as the number of messages in each VL.

A similar problem with the same solution space can be analyzed to see the number of possibilities. Grouping messages into VLs is equivalent to determining the number of distinct objects that can be placed into identical bins, where the number of identical bins is

anywhere from 1 to the number of objects. The solution to this problem is a summation of Stirling Numbers of the Second Kind [8].

$$\text{Permutations} = \sum_{i=1}^{\#\text{RDCs}} \sum_{k_i=1}^{n_i} \frac{1}{k_i!} \sum_{j=0}^{k_i} (-1)^{k_i - j} \binom{k_i}{j} j^{n_i}$$

where k_i is the number of virtual links for the RDC and n_i is the number of messages published from the RDC. As the number of RDCs of the network and number of messages published per each RDC increase, number of possible permutations is drastically increased. With the number of RDCs of a network usually between 10 and 20, and each RDC typically publishing at least hundreds of messages, this problem has a vast number of solutions. Thus, finding a close formed mathematical expression to find an optimal message grouping for a given network is nearly impossible.

2.2.3 Virtual Link Frame Structure

The frame structure used for VL frames closely resembles standard IEEE802.3 Ethernet frame structure [17]. Figure 2.7 below shows the structure for a VL [14]. The size in bytes is listed for each segment of the frame, with the maximum payload for each VL being 1471 bytes. If a payload ever exceeds 1471 bytes, its payload will be fragment into multiple transmissions. This is typically behavior that is avoided.

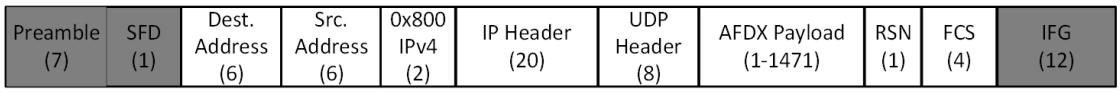

Figure 2.7: Virtual Link Frame

The differences from IEEE802.3 include the IP and UDP headers, along with the Received Sequence Number (RSN). Boxes highlighted in gray represent overhead due to the physical links. Boxes not highlighted in gray represent the VL's frame, where the AFDX payload is

variable. The overhead that is included for bandwidth calculation purposes includes every segment aside from the AFDX payload (67 bytes).

2.2.4 Bandwidth, Throughput, and Latency

Before looking at implementation of AFDX networks, an understanding of a few network definitions are needed. *Bandwidth* (B) is defined as the maximum rate of data transfer across a given path. In the avionics industry, this is typically a fraction of the possible total bandwidth of a physical link. Bandwidth is a constraint levied upon the network implementation based both on physical limitations, and leaving a margin to reduce jitter. For a full-duplex network there are two different aspects to bandwidth, source to destination and destination to source bandwidth. Since the physical connections for AFDX networks are bi-directional, both directions of the bandwidth must be analyzed. A given bandwidth is an upper bound for each physical link in the network, denoted by $B_{i,j}$ (where i, j represents the physical link from i to j).

$$\boldsymbol{B} = \{B_{1,2}, B_{1,3}, \ldots, B_{2,1}, B_{2,3}, \ldots, B_{i,j}\}$$

Latency (L) is defined as the time it takes for a bit of data to travel from a publisher to a subscriber. Evaluating the latency of each dataflow is necessary because latency impact is variable depending on the type of data that is being sent. For example, a engine failure message sent to a flight computer is very important and a high latency would not be acceptable. However an in-flight entertainment health message sent to the entertainment manager is not critical and a high latency would be non-ideal, but tolerable. Latency is measured per dataflow on the network. This is denoted by $L_{i,j}$ (where i, j represents the dataflow from i to j).

$$\boldsymbol{L} = \{L_{1,2}, L_{1,3}, \ldots, L_{2,1}, L_{2,3}, \ldots, L_{i,j}\}$$

Throughput (TP) is defined as the capacity of the bandwidth being used. This is measured as a percentage of the bandwidth used of the maximum bandwidth (as determined by the physical limitation and determined margin). A given throughput is the percentage of the maximum bandwidth that the physical link is using. Often times bandwidth and throughput are used interchangeably. Since bandwidth is a fixed constraint on the system, often times when researchers try to optimize bandwidth, in reality it is to optimize bandwidth usage (throughput). Throughput is denoted by $TP_{i,j}$ (where i, j represents the physical link from i to j).

$$\boldsymbol{TP} = \{TP_{1,2}, TP_{1,3}, \ldots, TP_{2,1}, TP_{2,3}, \ldots, TP_{i,j}\}$$

The following equation shows how to calculate the throughput for a particular physical link from (x to y) which has n number of VLs on it (where l_i is the frame size for the VL) [1].

$$TP_{x,y} = \sum_{i=0}^{n} \frac{l_i}{BAG_i}$$

Note that in order for the network to be feasible, the throughput must be less than the bandwidth for each physical link. If for some reason the calculated throughput is greater than the available bandwidth for the physical link, the VL structure will need to be recreated or a new routing scheme will need to be implemented.

2.2.5 Virtual Link Constraints

Because the VLs are harmonic (due to the BAG of each VL being a power of two), constraints can be derived via utilization analysis that form requirements levied upon BAG, MTU, and message grouping. These requirements will shape the allowable BAG (bytes) and MTU (ms) values.

The below expression, derived in research by [2], shows the message constraint of VL_i with n_i messages, each of size l_i (bytes) with a publish rate of p_i (ms) to guarantee the real-time requirement of all message flows (from i to j) in the link:

$$\sum_{j \in F_i}^{n_i} \frac{|l_j / MTU_i|}{p_j} \leq \frac{1}{BAG_i}$$

Similarly the following bandwidth constraint, derived in research by [2], for a system with

n VLs and a maximum bandwidth of B. Each VL_i is configured with $\{MTU_i, BAG_i\}$ such that MTU_i bytes are transmitted every BAG_i ms. Thus the $\{MTU_i, BAG_i\}$ for each VL_i must be configured to satisfy this constraint on bandwidth. This constraint is to ensure that the sum of all of the individual VL's bandwidths do not exceed the maximum bandwidth of the physical link. Note that each VL requires an overhead of 67 bytes. For more information on the overhead of each VL, see Section 2.2.3. Additionally, the factors of 8 and 10^3 are because B is measured as bits per second rather than bytes per millisecond.

$$8 \sum_{i=1}^{n} \frac{MTU_i+67}{BAG_i} \times 10^3 \leq B$$

Finally a jitter constraint, derived in research by [2], is levied upon the MTU choice. The ARINC 664-Part 7 specification requires jitter less than 500 μsec, with typical jitter in Ethernet hardware being 40 μsec [1] [2]. Thus a suitable MTU must be chosen for each VL such that the jitter constraint is satisfied.

$$40 + \frac{8 \sum_{i=1}^{n} (67+MTU_i)}{B} \leq 500$$

In a given network, the designers of end systems will inform the network engineers how often messages are published, as well as the size of the messages being sent (l_i and p_i from the equations above). Similarly, the designers of end systems work with the network engineers to identify how frequently messages need to be received. This defines the ultimate publisher and ultimate subscriber of the data. There are generally a few switches and other end systems which will pass the data along the AFDX network. How the data is packed into VLs and traversed through the network is only relevant to the network engineers as the end systems aren't affected by the path of the data. End systems only care that the data arrives on time. Packing the data into VLs and creating the path of the data through the network is where the majority of the design decisions are made in the AFDX network. In order to understand basic AFDX networking concepts, the definitions of a Remote Data Concentrator and a Remote Gateway are needed.

2.3 Remote Data Concentrator and Remote Gateway

A *Remote Data Concentrator* (RDC) serves the purpose of translating messages to and from avionics protocols (Analog to ARINC 664, ARINC 664 to Discrete, etc.). There is typically only one RDC per dataflow from publisher to subscriber, however there are instances where an RDC might send a message to another RDC prior to sending the message to the ultimate subscriber.

The *Remote Gateway* (RGW) is the logical side of the RDC. This means it is a hosted software application used to configure gateways to allow the hardware to send and receive data. A RGW controls the input and output data (I/O) for the RDC. There could be many RGWs per each RDC, but each RGW will be tied to a single RDC. Each RGW is configurable to define the messages received, the way messages are constructed, and how messages are published to subscribers.

For a typical network there will be end systems publishing and subscribing to data, however the protocol that the end systems can communicate through could be any viable avionics protocol. For example let's say an end system is publishing a message in ARINC 664, and there are two subscribers to that message expecting to receive it in ARINC 664.

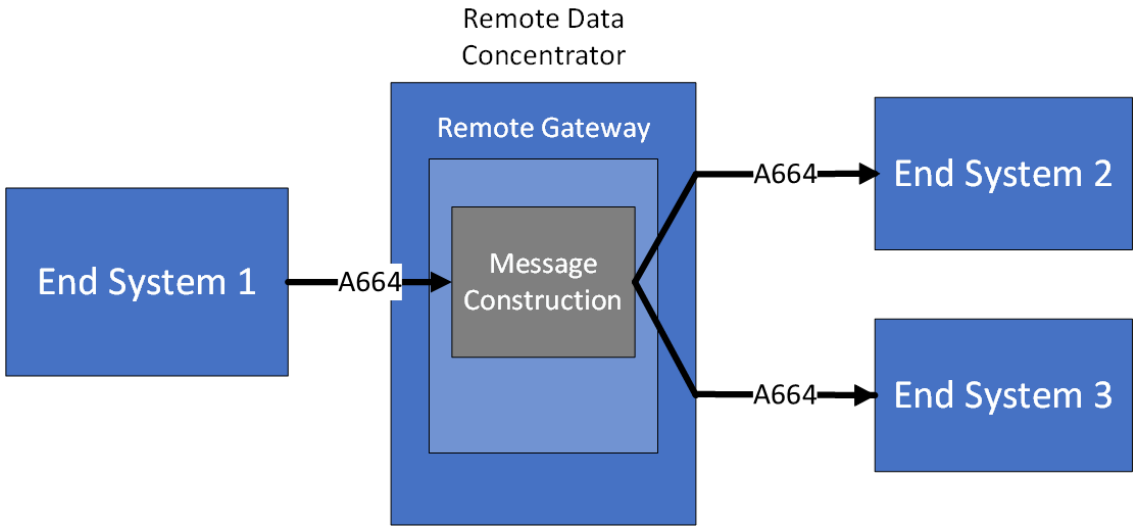

Figure 2.8: Simple Network Example

The following would occur at the logical layer in Figure 2.8:

1. Publish message (A664)
2. A664 Message sent to Remote Gateway on Remote Data Concentrator
3. Message reconstructed as A664 message
4. Message sent to subscribers

This shows messages are sent between the RGW and the two ultimate subscribing end systems. The way the data is carried between the end systems is through a virtual link.

2.4 Interface Control Document

All of the logical layer information is contained in the *Interface Control Document* (ICD) which maps out the avionics network for the entire aircraft. The ICD consists of XML files which include all the detailed information of the AFDX network. ICDs are used to define the physical layer (such as Flight Management Computers, Landing Gear, In-flight Entertainment Systems), as well as the logical layer (defining the dataflows between physical units). The ICD creates a mapping between the physical and logical layer, hence the entire AFDX network is captured within the ICD.

Once an ICD has been created, builds are done on the ICD to create software which is loaded onto the physical units of the AFDX network. The ICD is constantly being updated during network development. However once a solution is finalized, the ICD is not updated and deployed to the field. Prior to deploying to the field, simulations are run to gather network performance. The two performance characteristics that this research will focus on are latency and throughput numbers for the network.

The ideal network would have no latency, no jitter, and would have a throughput that is 100% (perfectly optimizing the physical network). This is impossible. A more realistic end goal for a network is to have minimal latency, minimal jitter, and would have a throughput that leaves enough margin such that contention is minimized and there is room for growth.

CHAPTER 3

CURRENT DESIGN APPROACH AND RELATED WORK

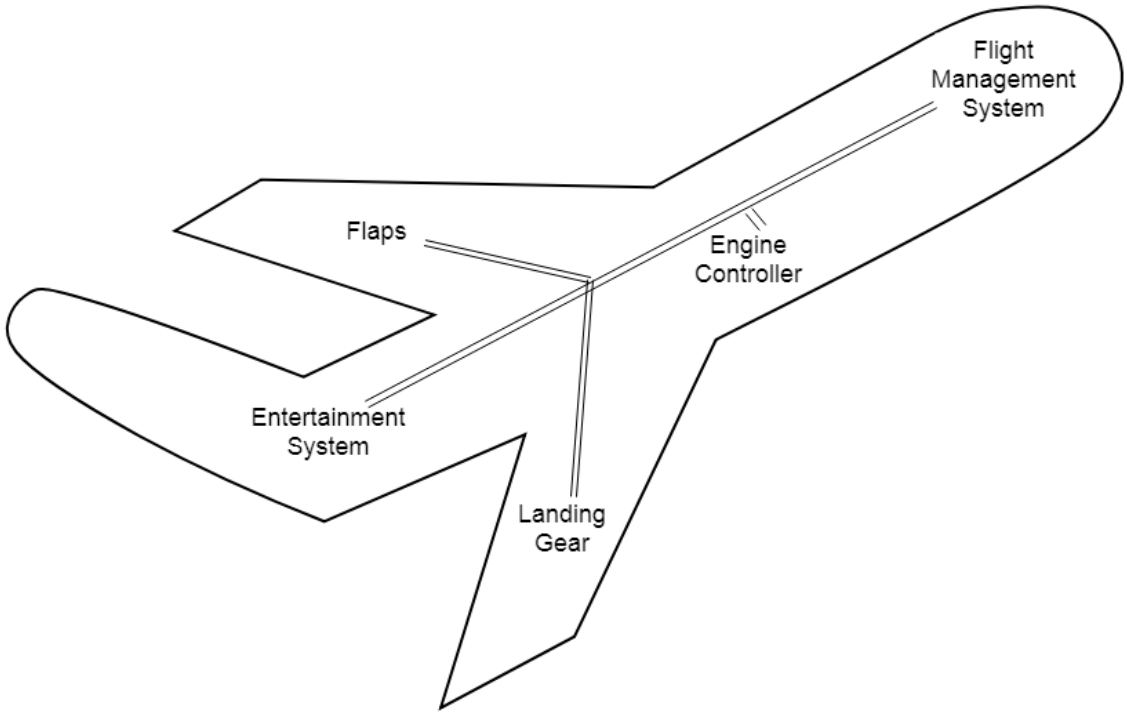

Figure 3.1: Example Aircraft AFDX Network

Each of the systems in Figure 3.1 represent an ES which needs to communicate with the AFDX network. The location of these end systems are determined prior to the network's design. The network is created by determining which of the end systems need to communicate and designing the road map of how data will travel to and from each end system. Messages with similar paths can be then grouped into virtual links, which define the path from the publisher to the subscriber for each message. For example, if the aircraft's Flight Management System publishes a message from both the flight data recorder and engine controller of the airplane, they might be grouped together in a virtual link and sent from the Flight Management System to those subscribing end systems. That virtual link would have an associated BAG and MTU, and possibly other messages in that share a similar path.

The current approach for the commercial aviation networking area is to only evaluate network statistics after the network has already been designed, with a few lessons learned from previous programs implemented in the design. When latency issues arise, for example, network redesigns are required. These redesigns could be as simple rerouting messages though different hardware, or as serious as requiring major scheduling changes to the end systems messages. AFDX networks nearly always meet the requirements levied upon the system by the aircraft manufacturer, however there is much room for improvement in the latency and throughput of the system. Designing the optimal solution for these two will ensure the best possible network solution is found for a given physical layout. The problem is not trivial because a network design is trying to balance the networks characteristics. For example, trying to decrease the latency will likely increase the throughput. Determining the best possible network design is a balancing act and can change depending on what the stakeholders are interested in and what size bandwidth margins are deemed acceptable.

When looking at possible changes to improve latency and throughput there are a few ways to approach a solution. One solution would be to move the end systems around to reduce physical travel time. While this solution would be the most simple to implement from a software perspective, often times the physical locations of end systems are determined prior to network design. The other way to approach it is to make changes at the logical level to adjust pathing and scheduling characteristics. This research focuses on the logical layer, more specifically manipulating virtual links and scheduling, while assuming that the hardware is constant.

3.1 Optimal BAG and MTU Values

Two parameters on virtual links that can be manipulated are BAG and MTU. Research has been done for determining the optimal values for MTU and BAG values while still adhering to the constraints noted by Section 2.2.5. The approach taken by this research group is to determine every possible BAG and MTU values for a given set of VLs. Then

use a branch and bound technique to determine the optimal values for the VL. Figure 3.2 shows the results of the two branch and bound algorithms used and the results of a brute force technique [2].

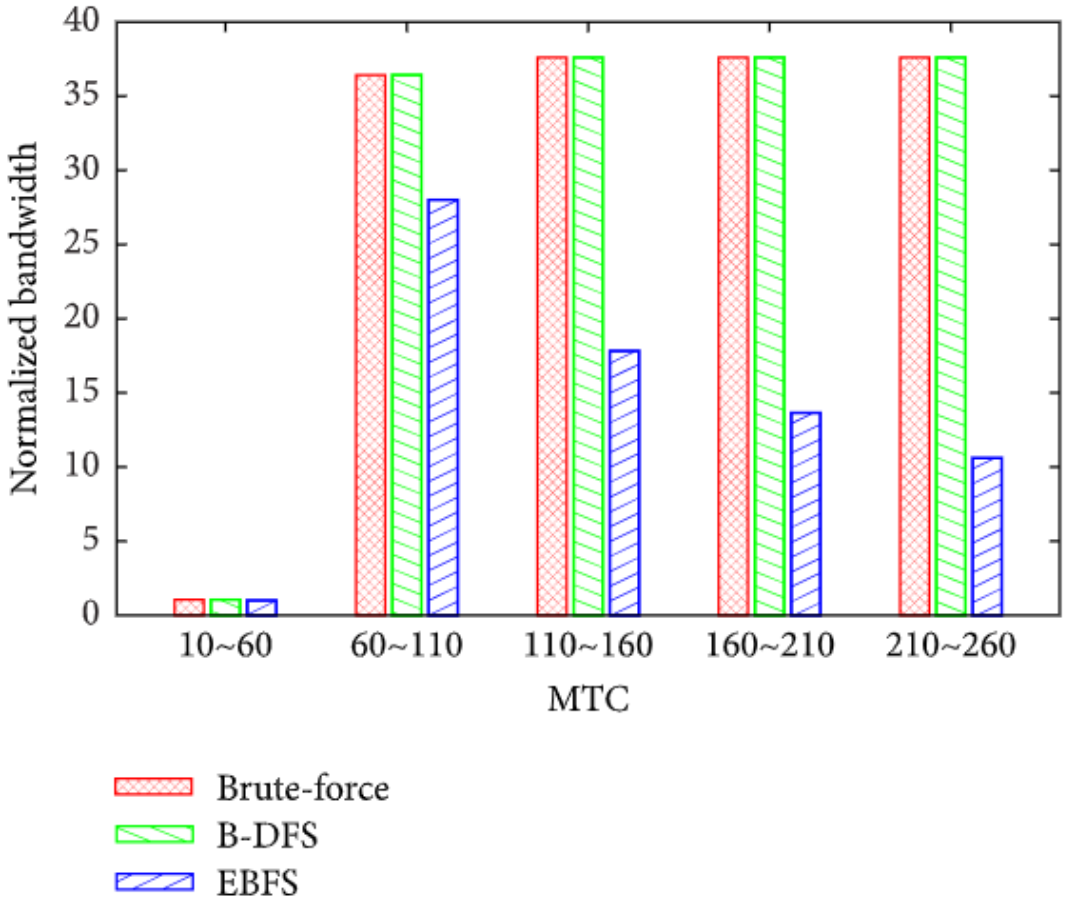

Figure 3.2: Branch and Bound Results [2]

While the results from Figure 3.2 shows an improvement for selecting a BAG/MTU suitable for a given set of VLs, it doesn't address the issue of how messages are grouped into VLs. The test run in [2] with only 6 virtual links, which is a small number in comparison to AFDX networks for a full aircraft. A more realistic test scenario would have anywhere from 100 to 1000 VLs [25].

3.2 Message Grouping and Virtual Link Aggregation

Another area to manipulate for optimization is the grouping of messages into VLs and sub-VLs. Research has been done to determine the optimal grouping of messages into Virtual Links to reduce bandwidth [4]. As part of the research in [4], they used two algorithms to grouping the messages. Algorithm 1 is to start with one large VL and start splitting into separate VLs when the bandwidth would decrease as a result of the split (noted by the first row of the table in section 2.2.1). Algorithm 2 is to have each message in its own VL (noted by the last row of the table in section 2.2.1), then combine VLs when the bandwidth would decrease as a result of combining. The results of this research is summarized by Figure 3.3. Figure 3.3 shows the network's bandwidth for all messages in their own VL, all messages in 1 VL, and the two algorithms stated above [4]. For all three test data sets, Algorithm 2 performed the best in reducing the network's bandwidth.

The algorithm for splitting and combining is a greedy algorithm and in likelihood would settle on a local minimum as opposed to the global minimum. This test was used with a set of 8 messages. This is a very small number for AFDX networks, messages are typically in the hundreds or thousands per RDC [25]. Due to the small number of messages, the possibility of having too many messages grouped into one VL is not an issue. However in real networks, this would be an issue because of the 1471 byte payload limit noted in section 2.2.3. Also this doesn't address how the BAG and MTU are recalculated when a new grouping scheme is created. Finally the number of VLs doesn't take into account the affect of latency on the system as a result of aggregating or splitting of VLs. If the number of VLs that an ES publishes is only 8, such as in [4], there will be no difficulty in scheduling and transmission of those VLs. When the number of VLs increases, scheduling and transmission become more difficult which leads to an increase in latency. Because the test data set is small and latency is not evaluated, this is a gap in research that needs to be investigated.

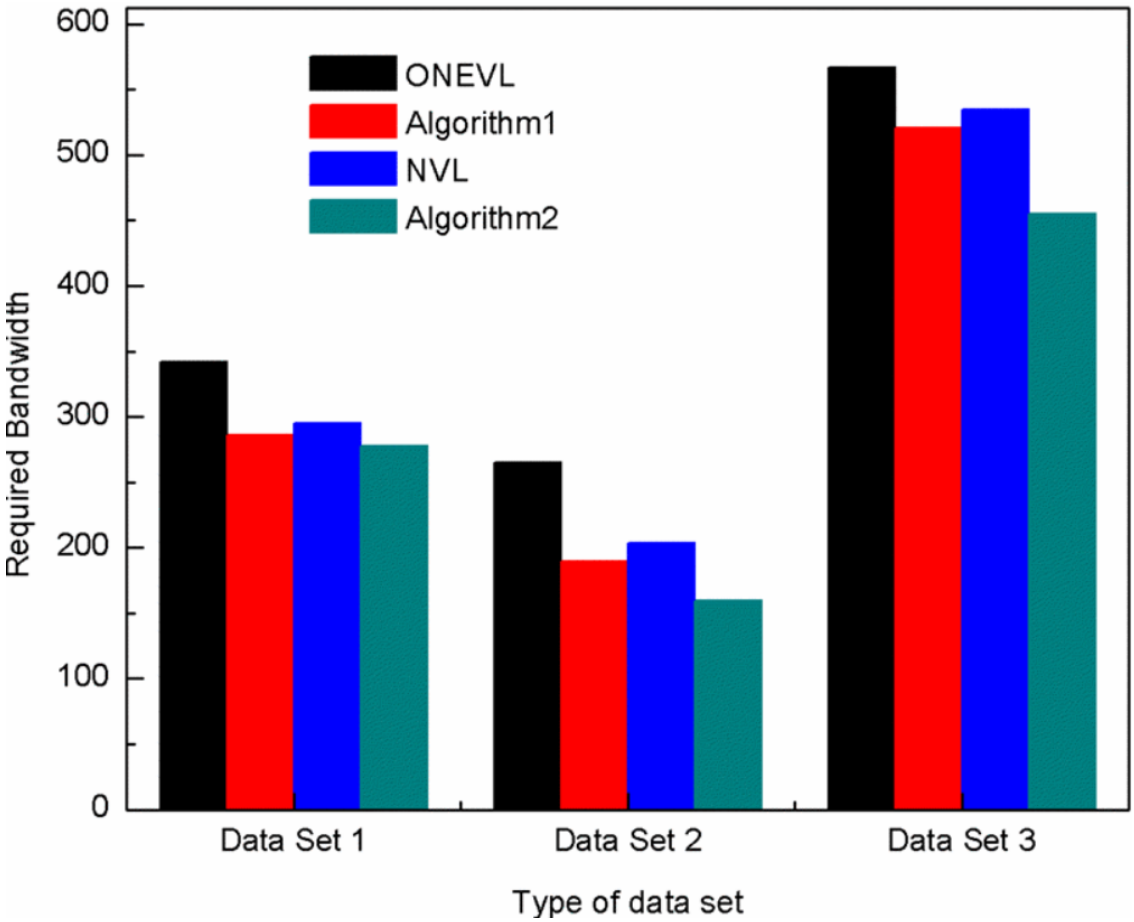

Figure 3.3: Message Grouping Results [4]

3.3 Optimized Transmission Scheduling

Another potential solution for network optimization is a more robust scheduling algorithm for transmitting VLs. Research has been done to evaluate the performance of algorithms for scheduling based on the following characteristics [3]:

1. Round Robin - a policy to allow the data frames of all queues, i.e. all virtual links in this case, to be regulated in each turn. During each round of the operation, all virtual links are guaranteed to be equally served one by one. The aim of Round Robin is to provide fairness of all virtual links regardless their arrival time, queue size, etc.

2. Smallest BAG - a policy to allow the data frame of the virtual link of which the BAG

value is the smallest to be regulated before the other data frame regardless their arrival time at the queue.

3. Longest Queue - a policy aimed to reduce the queue size as quickly as possible to avoid frame dropping due to no space left in the queue for a new coming frame. This policy allows a data frame of the virtual link, of which the number of frames waiting in the queue for traffic shaping is the largest, to be regulated before the others. This policy, hence, is suitable for the dense traffic communication where the frame dropping rate is high.

4. Smallest Size - a policy allows the smallest data frame to be regulated before the other larger data frame regardless their arrival time at the queue.

5. FIFO - The FIFO allows the data frame to be regulated according to their arrival time at the queue. The data frame which arrives first will be regulated before the others. This is a similar approach to a more comprehensive study on FIFO policy in AFDX networks [22] [23].

The results of the research can be seen in Figure 3.4.

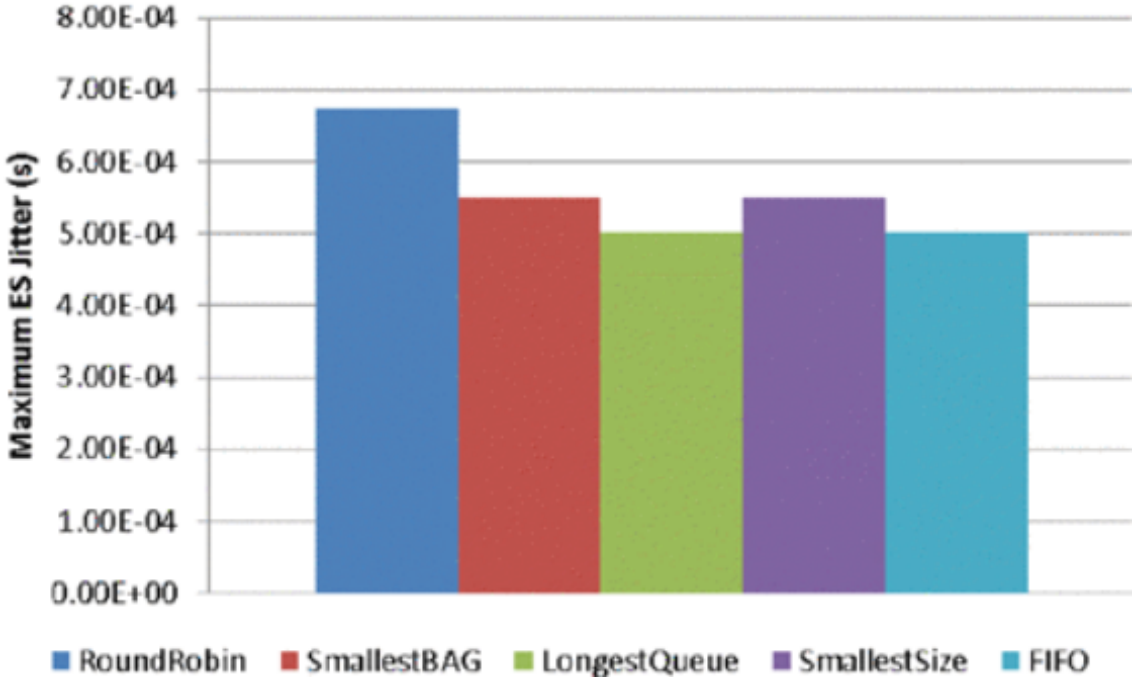

Figure 3.4: Scheduling Algorithm Results [3]

The goal in the research of [3] was to evaluate the algorithms based on the lowest jitter for the system. The results of their research indicated that a FIFO and Longest Queue algorithm were the best performing scheduling algorithms. This result shows how to reduce jitter, however this test was run on a low loaded network. A more comprehensive test would be needed to ensure these results hold true for a realistic AFDX end system.

3.4 New Contribution For Research

The discussed algorithms in this chapter discuss different optimization techniques that have been tried for AFDX networks. After an extensive search of research in this field, a customizable algorithm that accounts for both latency and bandwidth does not exist. Also Particle Swarm Optimization has not been attempted

CHAPTER 4

PARTICLE SWARM IMPLEMENTATION

The following sections will lay out a unique solution to solving AFDX network optimization. The provided algorithm in this chapter will be configurable such that the swarm will converge on a solution that is the best for the stakeholders (whether it's low latency, low bandwidth or a custom combination of both).

4.1 Particle Swarm Optimization Introduction

Particle Swarm Optimization (PSO) is an non-deterministic, meta-heuristic, optimization algorithm originally developed by Kennedy and Eberhart [6] [7] [18]. It is classified as an evolutionary algorithm which means that over iterations, the position of particles will shift towards the best performing particle of the group. Over enough iterations, the particles will converge on a good solution. A visual representation of this is shown in Figure 4.1.

The particles are initialized at random x values on the graph to start. Each x is an N-dimensional vector, where N is the dimension of the optimization problem that is being solved. Each of the particles is evaluated at their current position. After evaluation, each particle's x vector is pushed toward the particle with the minimum F(x) value of the swarm. After the particles are moved, the next iteration of particles are evaluated at their new positions. After evaluation, each particle's x vector is again pushed toward the particle with the minimum F(x) value of the swarm. Note that this could be a different particle than the first iteration. After enough iterations, the particles will converge onto a minimum solution. For a simple function similar to the one noted in Figure 4.1, the global minimum will most likely be found. However for a very complex N-dimensional solution, the particles might converge on a local minimum. While the algorithm might settle on a local minimum, this still might be a better solution than found without using this algorithm. With more randomness in the position update functions and a sophisticated particle update function,

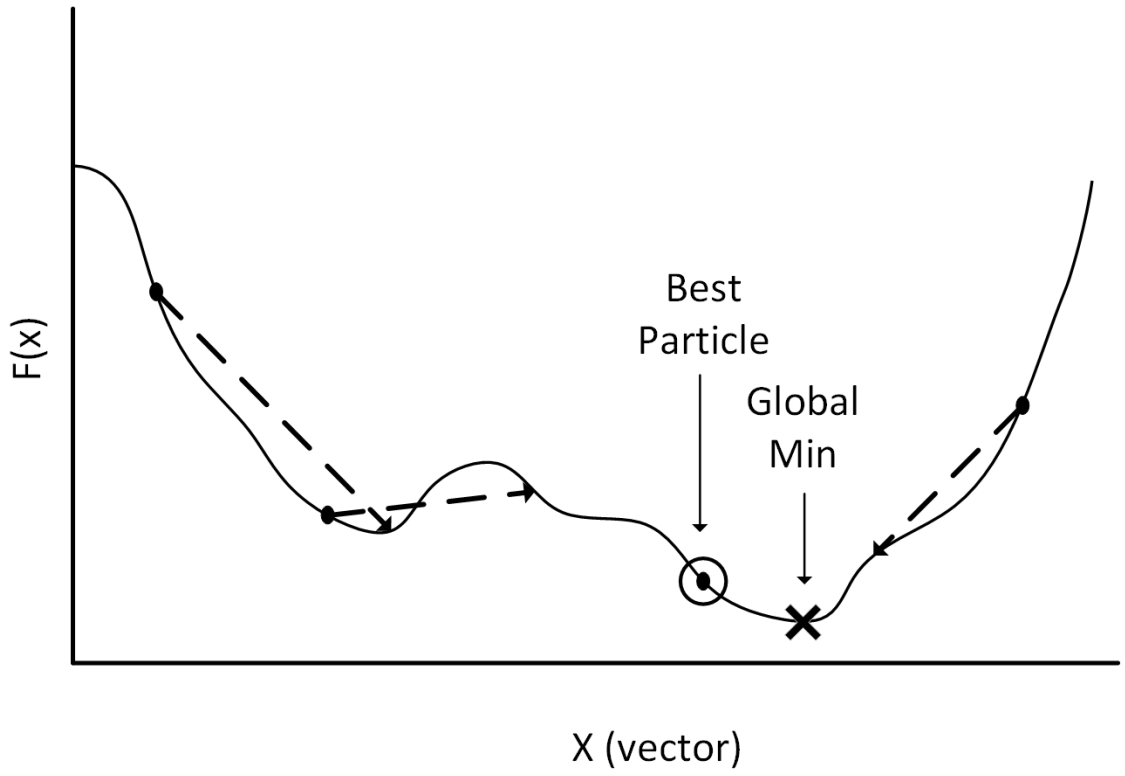

Figure 4.1: Particle Swarm Optimization Example

there is a better chance of converging on a good solution.

4.2 Particle Swam Optimization Mathematics

Each particle is defined by its position and velocity. Consider the j^{th} particle from the i^{th} iteration in an N dimensional space. It would be represented by the following position and velocity tuples:

$$x_j(i) = \{x_{j1}, x_{j2}, ..., x_{jn}\}$$
$$v_j(i) = \{v_{j1}, v_{j2}, ..., v_{jn}\}$$

At any particular iteration, each particle will have its own position and velocity. Both of these tuples are a function of the iteration, with the position update function defined as:

$$x_j(i+1) = x_j(i) + v_j(i+1)$$

The hyperparameters which define the behavior of the velocity function of the swarm are m, c_1, and c_2. The multiplier for the carry over velocity, $v_c(i)$, from the last iteration is given by m. Parameter c_1 gives the cognitive multiplier, while c_2 gives the social multiplier. The cognitive velocity, $v_{\text{cog}}(i)$, takes the difference between the particle's own best position, regardless of iteration (y_j), and it's current position. The social velocity, $v_{\text{soc}}(i)$ takes the difference in position between the particle's current position, and the position of the best particle in the swarm at the current iteration ($\hat{y}_j(i)$). The velocity function is defined as:

$$v_j(i+1) = v_c(i) + v_{cog}(i) + v_{soc}(i)$$

$$v_c(i) = m \times v_j(i)$$

$$v_{\text{cog}}(i) = c_1 \times r_{1,j}(i) \times [y_j - x_j(i)]$$

$$v_{\text{soc}}(i) = c_2 \times r_{2,j}(i) \times [\hat{y}_j(i) - x_j(i)]$$

PSO is non-deterministic due to the particles being initialized randomly to start and the update functions not being constant. Research has shown that introducing a bit of randomness in the velocity update function with clamping yields a faster convergence [9]. This is where the $r_{1,j}(i)$ and $r_{2,j}(i)$ functions come in. The functions $r_{1,j}(i)$ and $r_{2,j}(i)$ generate a random vector (with each value in the vector varying from 0 to 1) for each particle. These random vectors are multiplied by the velocity vector to get the final particle velocity for the iteration.

As an example, imagine the hyperparameters were set to $m = 0, c_1 = 0, c_2 = 1$. Without the random multiplier on the social velocity, each particle of the swarm would jump to the best particle of the first iteration and there would be no change in subsequent iterations. Another similar example where $m = 1, c_1 = 0, c_2 = 1$ would result in the swarm jumping to the best particle of the first iteration, then performing an equal jump past the group's best during the second iteration since the carryover velocity is set to 1. This choice of hyperparameters would struggle to settle on a minimum due to this un-damped oscillation. Using the intuition from these two examples, it is easy to see that hyperparameters are

typically chosen to be between 0 and 1.

4.3 Modeling Message to Virtual Link Assignment

PSO is the method that will be used to find the best message to VL assignment. In order to implement this, a particle structure is needed to translate how a particle will represent a message grouping configuration. Each particle will be represented by a matrix. The example particle in Table 4.1 would represent a particle for 3 RDCs publishing 4, 5 and 3 messages respectively:

RDC Messages	Msg 1 Group	Msg 2 Group	Msg 3 Group	Msg 4 Group	Msg 5 Group
4	1	3	3	4	X
5	3	2	1	3	4
3	2	2	2	X	X

Table 4.1: Example Particle

Each row in Table 4.1 represents one RDC. The first column of each row is the total number of messages published by that RDC. The rest of the columns in that row will be constrained to be an integer between 1 and the number of messages transmitted by that RDC. Two cells in a given row having the same integer value means that they will be grouped together in a VL. If all of the cells in a given row have the same value, they will all be grouped into one VL. The example PSO particle in Table 4.1 equates to 3 VLs in RDC1 (messages 2 and 3 being grouped together), 4 VLs for RDC2 (messages 1 and 4 being grouped together), and 1 VL for RDC3 (messages 1 2 and 3 being grouped together).

Note that this particle modeling is only one of many possibilities. The discussed modeling in Table 4.1 has an enormous solution space if the number of messages transmitted for an RDC is in the hundreds or more. One alternate approach would be to limit the number of virtual links that the RDC could transmit. For example, if an RDC publishing 5 messages were limited to a maximum of 3 VLs. The constraints on the particles would be that the published messages would be assigned an integer from 1 to 3, as opposed to 1 to 5. An

example of the limited VL modeling can be shown in Table 4.2.

Messages	Msg 1 Group	Msg 2 Group	Msg 3 Group	Msg 4 Group	Msg 5 Group
5	1-3	1-3	1-3	1-3	1-3

Table 4.2: Example Particle With Limited Virtual Links

An additional way to model the problem to reduce the solution space would be to define a set number of VLs that must be non-empty. For example, if an RDC is publishing 100 messages must publish exactly 20 VLs. The constraints on the particles would be that the published messages would be assigned an integer from 1 to 20, as opposed to 1 to 100. Also there will be a constraint on the particles there is at least one message for every integer from 1 to 20.

4.4 Framework for Implementation

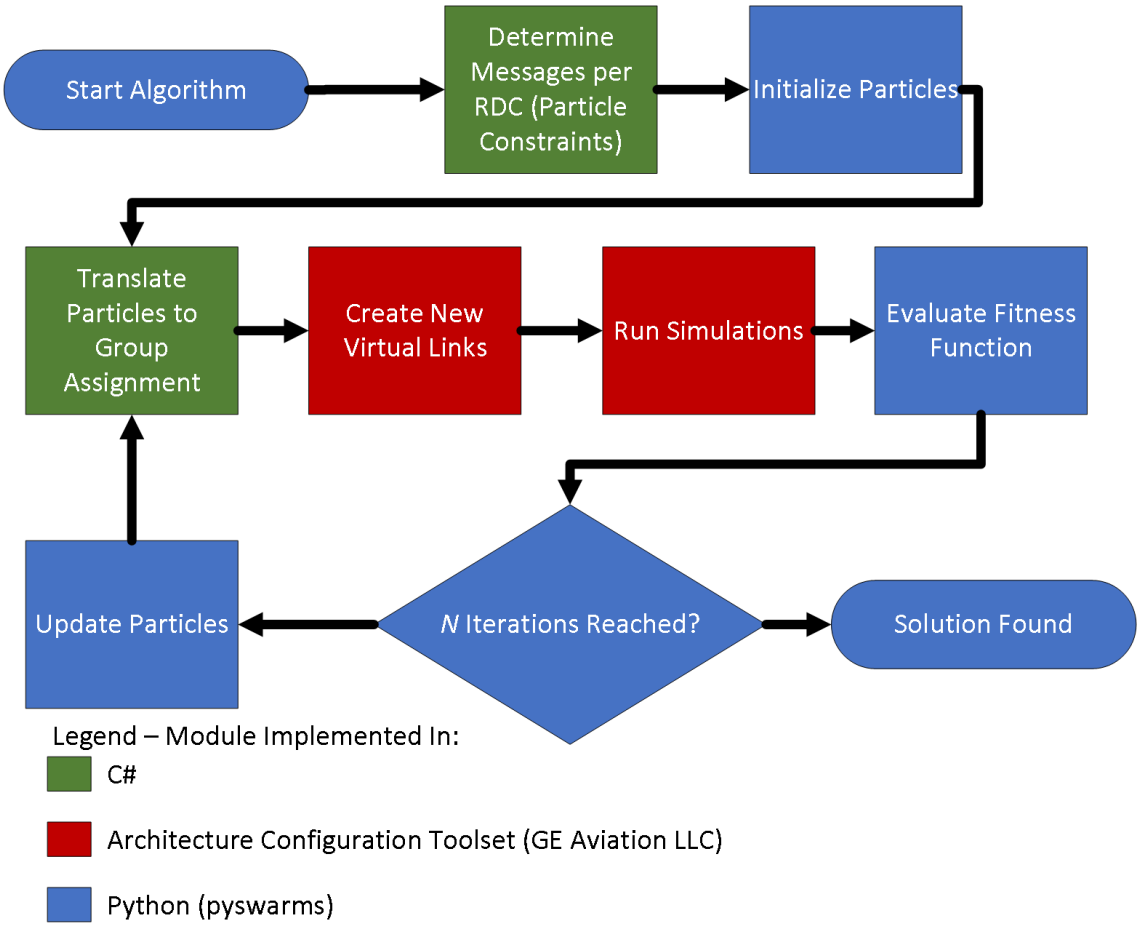

Figure 4.2: Optimization Block Diagram

Figure 4.2 shows the process for implementing this algorithm. Included in the figure is the software that was used to implement each portion of the framework. First, the network will be analyzed to determine number of messages published per RDC. This will define the constraints on the particles for optimization. Next, the particles will be randomly initialized to define their initial position in the solution space. Each particle will then be translated from a particle (defined as a matrix as shown in 4.3), to a VL grouping assignment in the network model. This is done via a parsing script to create an XML file, similar to the few lines shown in Figure 4.3. Once the XML file is created to define the groupings, a series of SQL queries are executed on the network model to implement the message to VL groupings.

```
<MsgVLGroup Name="RDCHW_1_USG267">
  <Group_Ref DestName="RGW1_A664_To_A664_10.Tx_A664_To_A664_EF_Port2" />
  <Group_Ref DestName="RGW1_A664_To_A664_16.Tx_A664_To_A664_EF_Port2" />
  <Group_Ref DestName="RGW1_A664_To_A664_21.Tx_A664_To_A664_EF_Port" />
  <Group_Ref DestName="RGW1_A664_To_A664_7.Tx_A664_To_A664_EF_Port" />
  <Group_Ref DestName="RGW1_A664_To_A664_9.Tx_A664_To_A664_EF_Port" />
</MsgVLGroup>
```

Figure 4.3: Example Grouping in XML

Now that the new groupings have been defined by the algorithm and translated into groupings in the network's SQL model, VLs can be created. VL creation is done using a deterministic algorithm to ensure that comparison between particles is relevant. With the newly created VLs (each with their own BAG and MTU values) implemented in the network model, the particle has been translated from the PSO algorithm to the network model. Next, simulations are executed to determine the network performance. After the simulation is completed, the results are pulled and analyzed to determine how successful the simulation was for optimizing the AFDX network.

After each particle in the iteration has been evaluated, the algorithm will push all the particle toward the best performing particle of the group. Once each particle has moved, the updated particles be translated into new virtual links and more simulations will be executed. After this process has completed for the set number of iterations, the solution will be found.

When implementing the PSO solution, the most important variable is the fitness function. The fitness function will define the "best" solution that is converged upon. The fitness function will shape the graph denoted in Figure 4.1. This means that the solution is configurable to converge on solutions that prioritize different characteristics of AFDX networks. To create the fitness function, the output of the simulation must be concatenated into one number, representing the fitness of the simulation. How the concatenation occurs is dependent upon the stakeholder's decision about the importance of each output of the network. For example, some stakeholders might be interested in extremely low latency but wouldn't care as much about high bandwidth utilization numbers. Other stakeholders might be interested in low bandwidth numbers while not caring about latency statistics. An example fitness function

(for a network with n dataflows and m physical links) would be the following:

$$F(x) = c_1 \times \sum_{n} \boldsymbol{L} + c_2 \times \sum_{m} \boldsymbol{B}$$

The stakeholders would be in charge of adjusting the weights of the simulation outputs to ensure a desirable solution is converged upon. With only the task of creating a fitness function to assign a single cost to each run, this algorithm allows for the stakeholder to customize a solution best suitable for their interests.

4.5 Scope of Algorithm

This algorithm is restricted to the logical layer of an AFDX network. This means that the physical layer for the network will be constant. From the logical layer, the following will be held constant

1. Message construction, rate, and path from the publisher to the RGW.

2. Message construction and rate from the RGW to the subscriber.

Figure 4.4 shows an example network for one RDC. The circled dataflows show the scope of where the message grouping will occur. Note that the message grouping will only be done on the messages published by the RDCs. This is because often times, publishing end systems will be predefined prior to the network's logical layer design. However this algorithm could be expanded to include messages published by end systems.

This isn't to say the the scope of algorithm couldn't grow to contain more than just what is listed here. To increase the scope of the algorithm, the particle definition would need to be updated to include the new scope. Increasing the scope would increase the time it takes for the solution to converge on a solution, as well as introduce complexity in how the particle would need to be designed. The algorithm, however, would still be feasible for the increased scope.

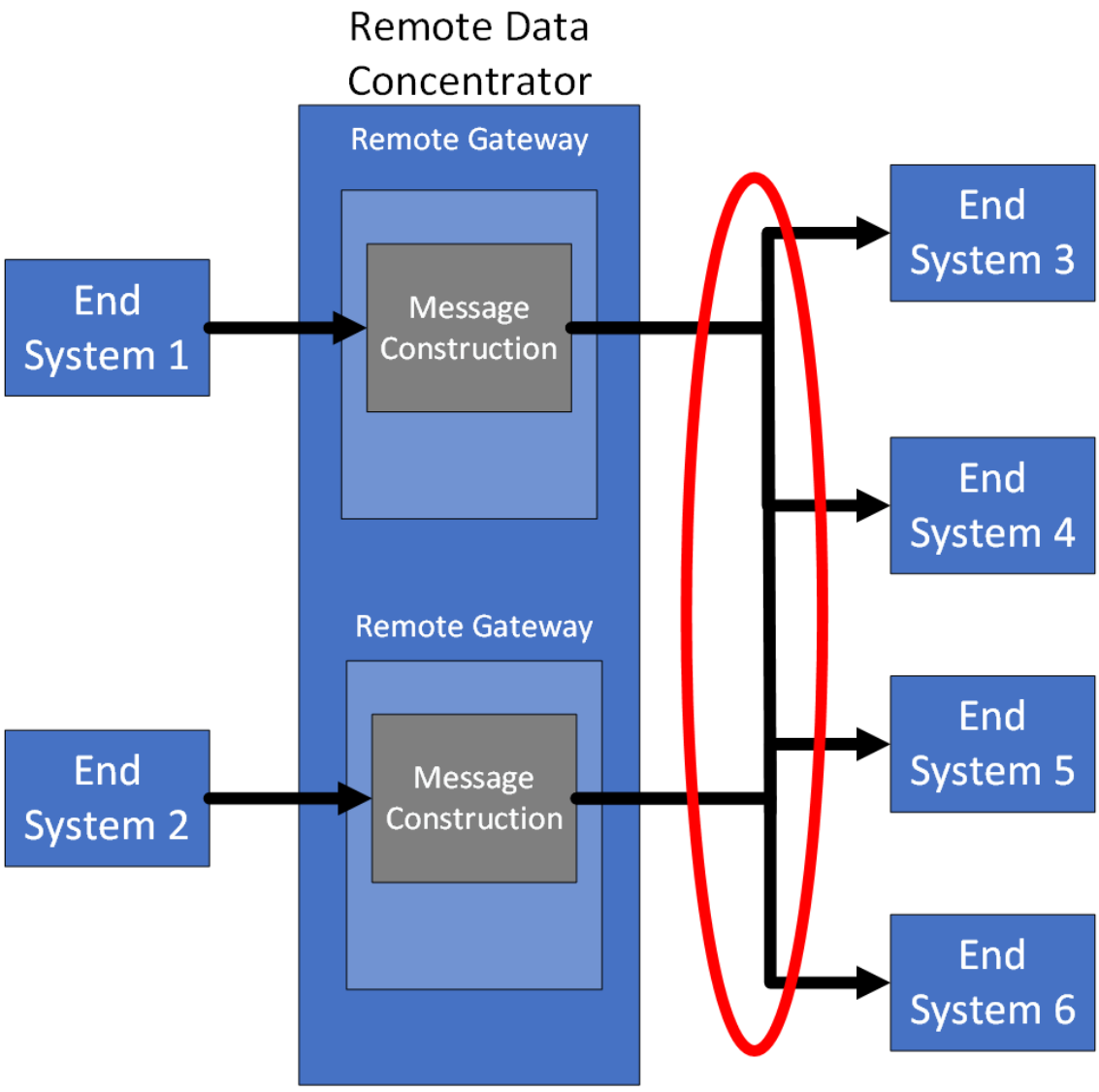

Figure 4.4: Scope of Algorithm

CHAPTER 5

NETWORK IMPLEMENTATION AND RESULTS

This chapter will go into detail about the network (test data) that was used for this research. An explanation of the hyperparameter choice will be provided. Finally, the results from a 3 week algorithm run will be shown to prove the algorithm is feasible for this optimization problem.

5.1 Network Used for Testing

The network design for testing included multiple publishing end systems, multiple subscribing end systems, one RDC, and switches. Figure 5.1 shows the setup for the network model for this research.

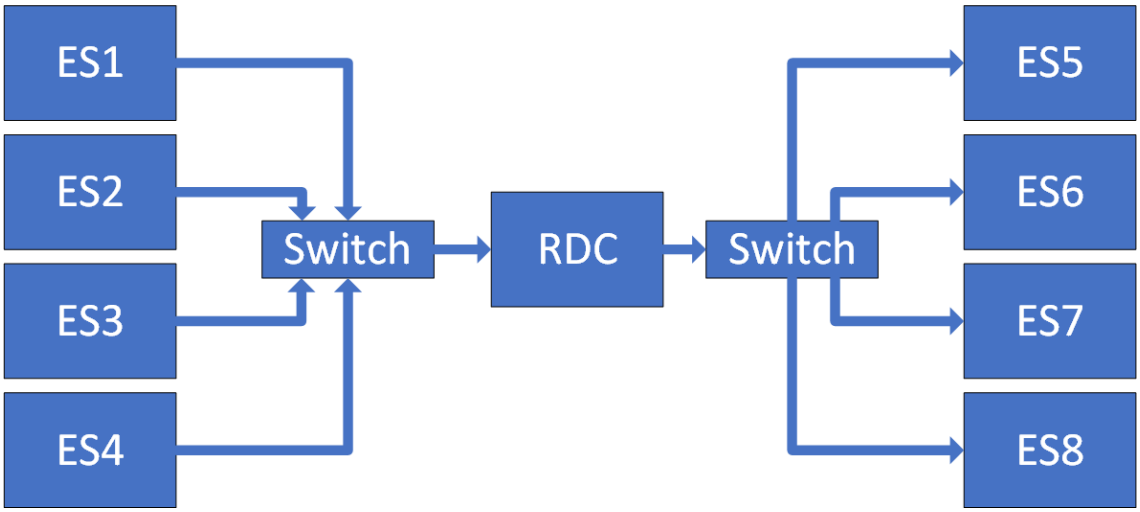

Figure 5.1: Test Network Design

The RDC for the network is publishing a total of 475 messages, varying in payload size from 16 bytes to 400 bytes. Each message is being published out at a rate of 20, 40, 80, 160, 320, 640, or 1280 ms. Note that both the number of messages and the size of the messages means that some grouping assignments are not possible due to the payload restriction noted in Section 2.2.3. When an illegal grouping assignment is created from a particle, a maximum

bandwidth (100%) is assigned to that particle. This will ensure that the other particles are not pushed towards this illegal grouping assignment.

This test data originated from a real aircraft network but was severely scaled down to save time and prove the feasibility of the algorithm. A typical real aircraft network would have around 4 - 20 switches, 6-20 RDCs, 200-500 end systems, and 1000-5000 messages.

The following computing hardware and software was used for testing.

- Hardware
  - Intel(R) Xeon(R) CPU E5-2640 @ 2.50 Ghz 2.50 Ghz (2 processors)
  - 32.0 GB of RAM
  - 64-bit Operating System
- Software
  - Python (2.6.5)
  - Python package - pyswarms (0.3.1) [5]
  - Architecture Configuration Toolset (GE Aviation Systems LLC)

5.2 Hyperparameter Selection

From the time the algorithm is started, each particle is initialized and a build is run to determine the performance of that particle. Each particle roughly takes about 20 minutes to complete. Once one particle is finished, the next particle is run. After the entire group of particles is completed in an iteration, the particles are updated per the velocity function noted in section 4.2. Because of the resource constraints and the time it takes to execute builds, five combinations of hyperparameters were tested for performance on a small number of particles and iterations to get a sense of how the algorithm would behave. The test patterns for the hyperparameters are listed in the table below [19]. Note that m is set to zero for this research. This is because during preliminary testing, particles were drifting

beyond the constraints discussed in 4.3. When particles drifted beyond the constraints, their velocity for the rest of the iterations was set to 0. This drove the decision to set m to zero for this research.

Test Pattern	m	c_1	c_2
1	0	.1	.9
2	0	.3	.7
3	0	.7	.7
4	0	.7	.3
5	0	.9	.1

Table 5.1: Test Patterns

Originally the fitness function included latency along with bandwidth. However in preliminary testing the latency, which averaged about 1250 ms, had a maximum variance of .25 ms between particles. This is more than likely because the physical ports on the RDC did not have enough traffic on them. When this variance was discovered, latency was no longer included in the fitness function. The updated fitness function is just taking the percentage bandwidth used for the physical link from the RDC to the switch connected to it. This is the data shown in the results that follow.

Note that the particles were not moving aggressively enough during initial testing. In an effort to speed up the convergence of the algorithm, the cognitive and social functions noted in section 4.2 were updated to use a random number from .5 to 1 instead of 0 to 1. While this compromises part of the randomness, this created a faster velocity which helped in a more aggressive algorithm to save on time.

Figure 5.2 shows the first test pattern. This has a low cognitive velocity and a high social velocity. This did not have a great improvement among the swarm as compared to the rest of the test patterns that follow. This highlights the importance of cognitive and social velocities together. This hyperparameter set will not be used for the final run.

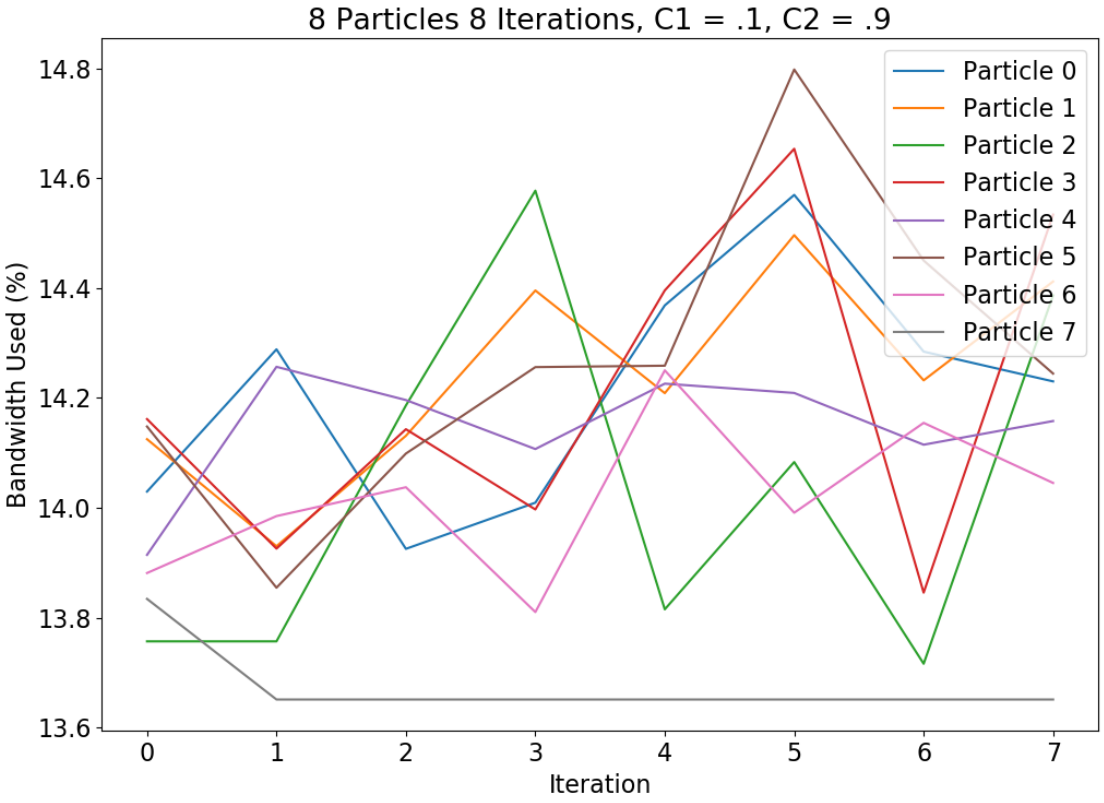

Figure 5.2: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .1, .9\}$

Figure 5.3 shows the second test pattern. This has a slightly higher cognitive velocity and a slightly lower social velocity. The trend for this test set is not desirable as their is not as much improvement as the next test pattern. Thus, this hyperparameter set will not be used for the final run.

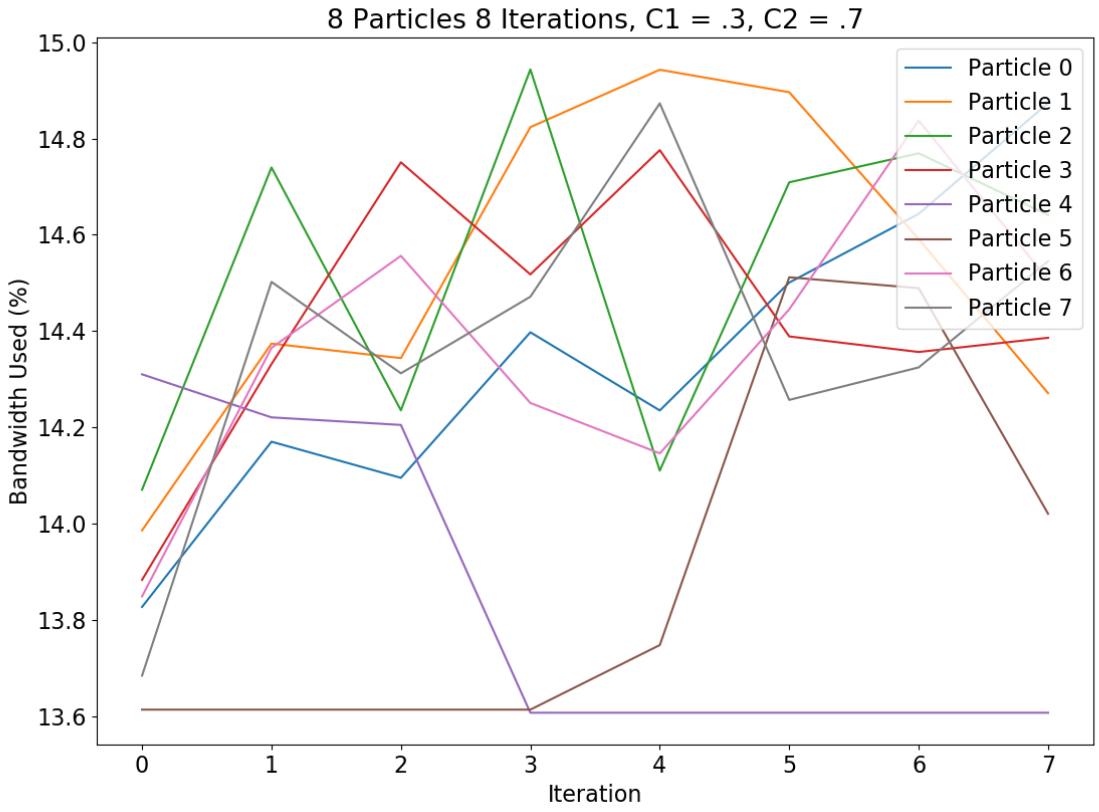

Figure 5.3: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .3, .7\}$

Figure 5.4 shows the third test pattern. This has an equal social velocity and cognitive velocity. Although this has the worst nominal result for the test sets, the trend of each particle is desirable since the improvement from the first to last iteration shows the best improvement. This is the hyperparameter set will be used for the final run.

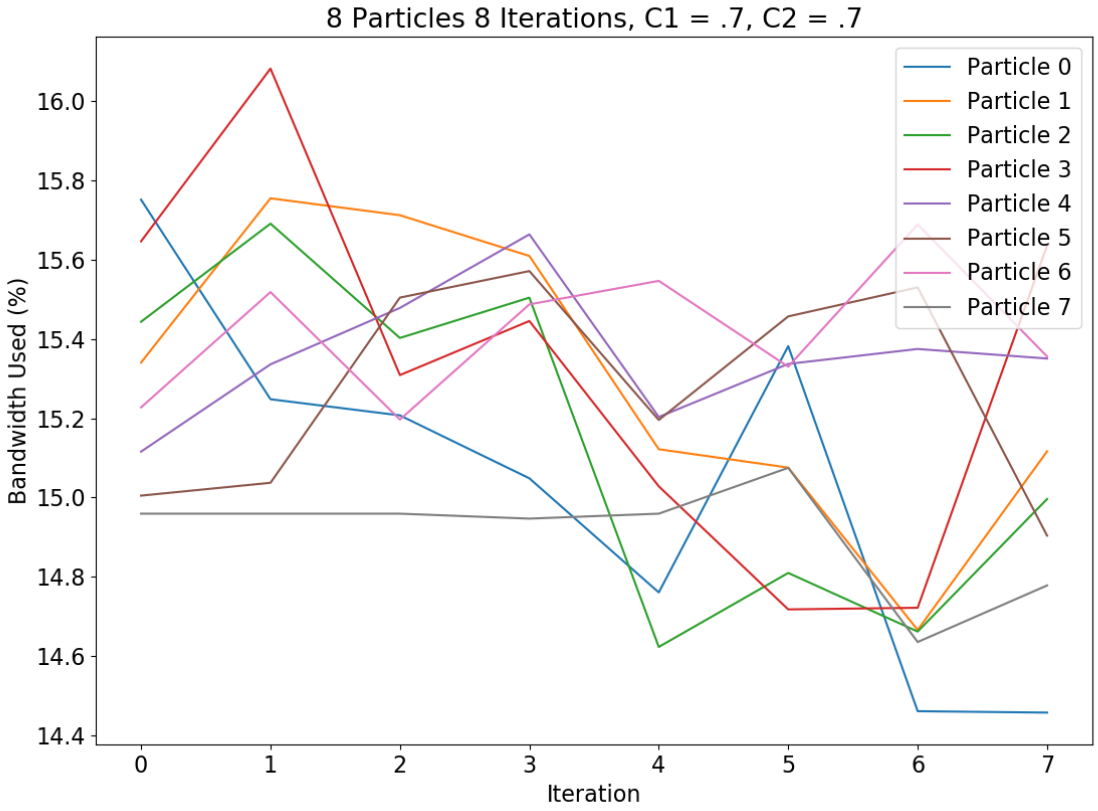

Figure 5.4: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .7, .7\}$

Figure 5.5 shows the fourth test pattern. This has a higher cognitive velocity and a lower social velocity. This yielded better results than test pattern two, but did not have as good of an improvement among the swarm as compared to test pattern three. This hyperparameter set will not be used for the final run.

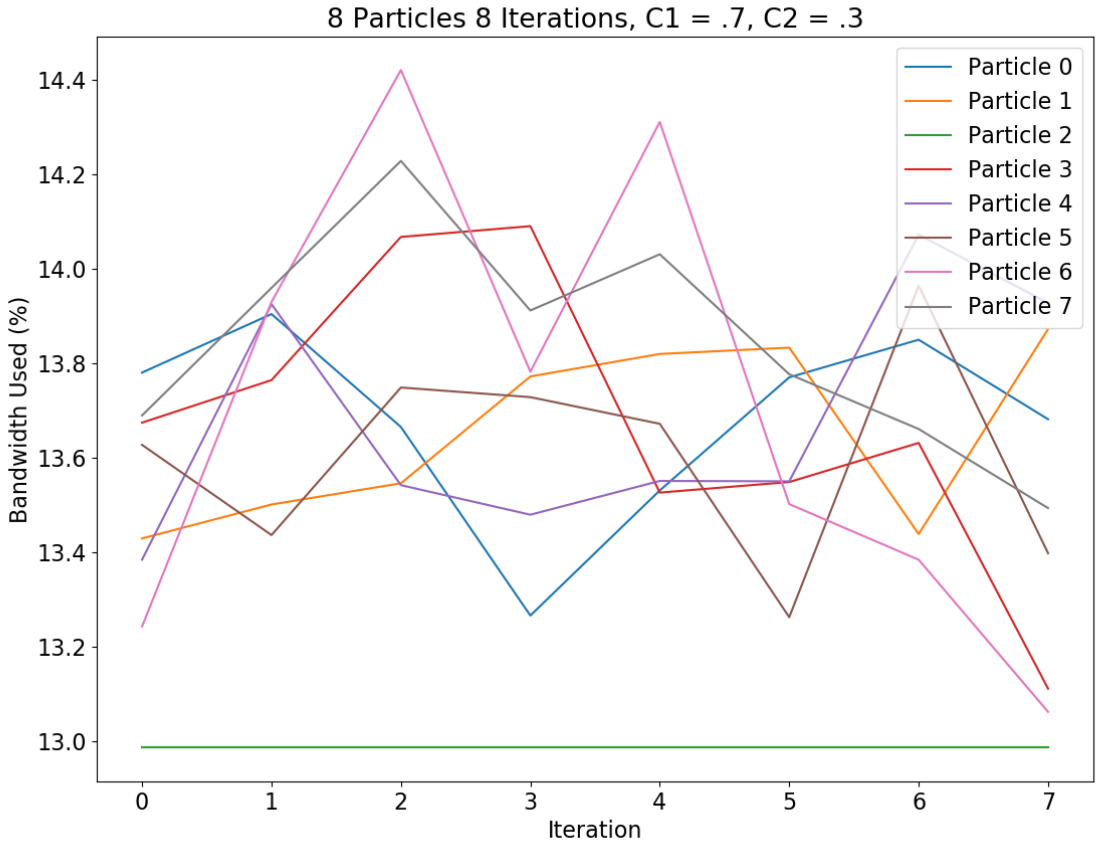

Figure 5.5: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .7, .3\}$

Figure 5.6 shows the fifth test pattern. This has a high cognitive velocity and a low social velocity. This trend for this test set is not desirable because in order to fully see the cognitive velocity's affect, there needs to be a great number of iterations to allow each particle to get a chance to move. This hyperparameter set will not be used for the final run.

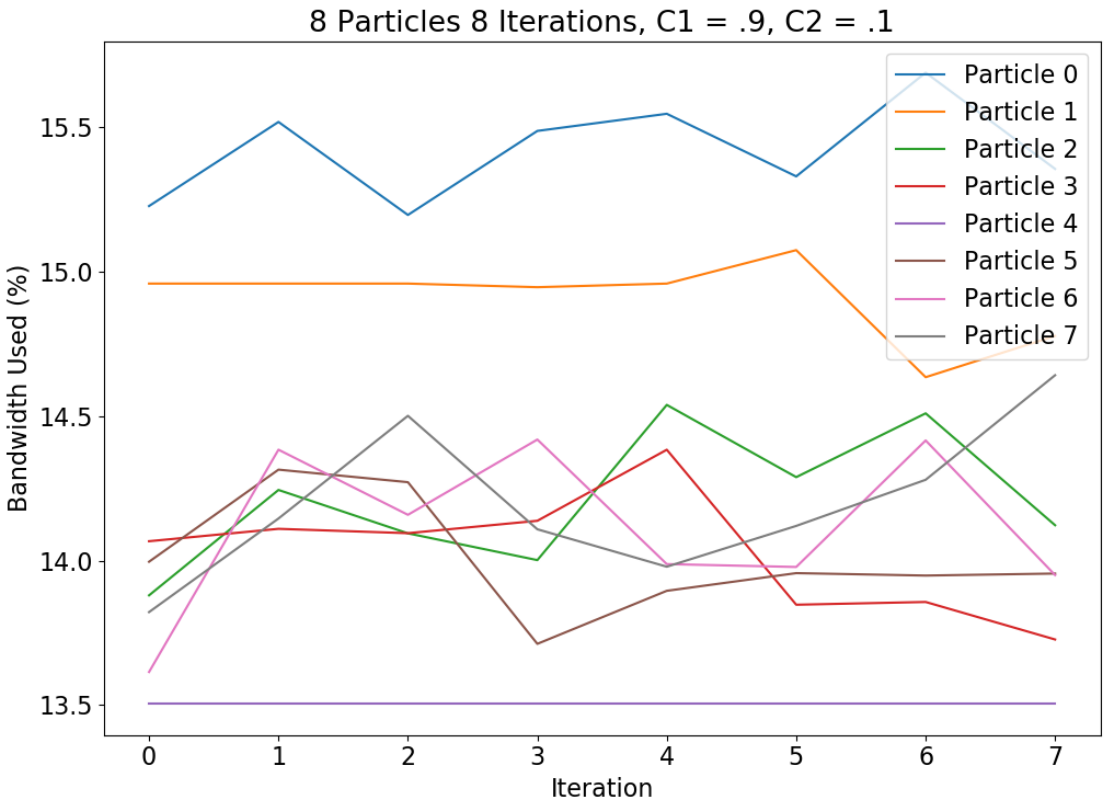

Figure 5.6: 8 Particles 8 Iterations $\{m, c_1, c_2\} = \{0, .9, .1\}$

5.3 Optimization Results

Figure 5.7 shows the final run with 10 particles over 100 iterations. Because each particle's evaluation takes about 20-30 minutes, this test setup took about 3 weeks to complete.

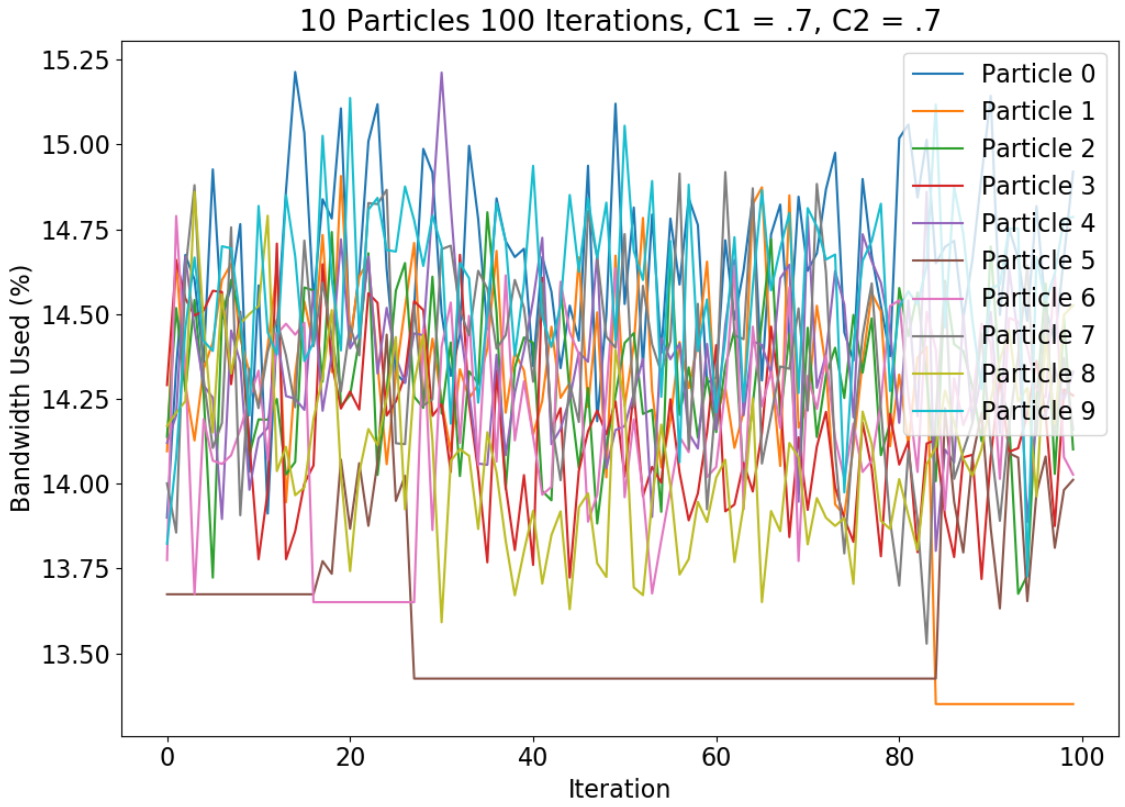

Figure 5.7: 10 Particles 100 Iterations $\{m, c_1, c_2\} = \{0, .7, .7\}$

Figure 5.7 shows a progression over time of a continuously improving bandwidth. Understanding that the algorithm was only run with 10 particles over 100 iterations, the slight improvement shown is expected. The important thing to see is that the same particle is not the global best for all iterations. A new particle becoming the global best shows that the particles are shifting and moving toward settling on a solution. If this algorithm would continue to run with a larger number of particles and more iterations, the algorithm would settle on a good VL grouping solution.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The contributions of this thesis include an explanation of the basics of the AFDX networks and the major factors engineers are interested in within the AFDX network. Additionally, a few researched solutions to AFDX network optimization were evaluated for effectiveness based on relevant criteria. It was discovered that an optimization algorithm that takes into account both bandwidth and latency had not been researched after an extensive review. To fill this gap, a framework for how to set up this optimization problem through Particle Swarm Optimization is detailed. While the algorithm was only used for bandwidth in this research, an explanation of how this algorithm could expand to include latency was detailed. The choice of hyperparameters for this research was $\{m, c_1, c_2\} = \{0, .7, .7\}$. Results for a sample swarm was shown to prove the feasibility and usefulness of this custom algorithm.

Ideally a comparison of the researched solutions would be presented to see how the Particle Swarm Optimization algorithm performs against the other algorithms on the same data set. Given time constraints and resource constraints, comparing the other algorithms against this data set was not possible for this research. Running the simulation to test each grouping scenario takes about 20 minutes. In order to implement the algorithms and gather results, it would take months to complete. Instead of implementing and testing the researched algorithms as a comparison, the limited resources were put solely into the implementation and testing of the Particle Swarm Optimization algorithm.

Future work would include running this algorithm over more particles and iterations. The limited resources for this research forced the research to turn more into a proof of concept. With more iterations and and more particles, the algorithm will converge onto a good solution. Another area of future work would be to run the different modeling technique described in section 4.3. The modeling that was used for this research, although feasible, has a large solution space. The different modeling techniques descried in section 4.3 attempt

to reduce the solution space. This would help the algorithm converge on a solution quicker than the original modeling technique used for this research. The final point for future works would be to put more data on the network to see more variance in latency from particle to particle. Because the latency variance was so small, the multi-objective fitness function's performance wasn't fully tested.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Aeronautical Radio INC, "ARINC Specification 664: Aircraft Data Network, Part 7 - Deterministic Networks", October 2003.

[2] Dongha An, Hyun Wook Jeon, Kyong Hoon Kim, and Ki-Il Kim, "A Feasible Configuration of AFDX Networks for Real-Time Flows in Avionics Systems", International Journal of Aerospace Engineering 2015:1-9, November 2015

[3] Suthaputchakun Chakkaphong, Lee Kin-Man-Benjamin, Sun Z. "Impact of End System scheduling policies on AFDX performance in avionic on-board data network." 1-6. IEEE 2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA), August 2015.

[4] Cha YoungJun, Lim JuHo, Lee SunYoung, Kim DongJin, Kim Ki-Il. "New feasible grouping algorithms for virtual link in AFDX networks." 1-4. IEEE 2015 International Conference on Informatics, Electronics & Vision (ICIEV). June 2015.

[5] Miranda L.J., (2018). PySwarms: a research toolkit for Particle Swarm Optimization in Python. Journal of Open Source Software, 3(21), 433,

[6] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization", Proceedings of the IEEE International Joint Conference on Neural Networks, 1995, pp. 1942-1948.

[7] R. C. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory", Proc. 6th Int. Symp. Micromach. Human Sci., pp. 39-43, 1995.

[8] D. E. Knuth, The Art of Computer Programming, Reading, MA:Addison Wesley, vol. I, 1969.

[9] Musaed Alhussein, Syed Irtaza Haider, "Improved Particle Swarm Optimization Based on Velocity Clamping and Particle Penalization", 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS), December 2015.

[10] Howard Courtney, "Data bus market to reach $8.55 billion by 2020 as aerospace system complexity grows" Intelligent Aerospace, 2015.

[11] C. Furse, R. Haupt, "Down to the wire" in Spectrum, IEEE, 2001.

[12] Kaufmann, M. Cost/weight optimization of aircraft structures. Licentiate thesis, KTH Engineering Sciences, Stockholm, Sweden. 2008

[13] Planespotters.net. American Airlines Fleet Details and History. 2019.

[14] Airbus, "Avionics Full Duplex Switched Ethernet Workshop", March 2010

[15] Software Considerations in Airborne Systems and Equipment Certification. RTCA document DO-178C, 2011.

[16] Li, M. "Determinism Enhancement and Reliability Assessment in Safety Critical AFDX Networks", École polytechnique de Montréal. Département de génie électrique, 2016.

[17] IEEE, "802.3-2018 - IEEE Standard for Ethernet", 2018.

[18] J. Kennedy, R. C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publisher, 2001.

[19] Shi, Y. and Eberhart, R. (1998) Parameter selection in particle swarm optimization. In Evolutionary Programming VIZ: Proc. EP98, New York: Springer Verlag pp. 591-600.

[20] A. Basu, S. Bensalem, M. Bozga, B. Delahaye, A. Legay, E. Sifakis, "Verification of an AFDX infrastructure using simulation and probabilities", Proc. 1st Int. Conf. Runtime Verification, vol. 6418, pp. 330-344, 2010, [online] Available: https://hal.archives-ouvertes.fr/hal-00557717.

[21] C. M. Fuchs, "The evolution of avionics networks from ARINC 429 to AFDX", Proc., pp. 65-76, 2012.

[22] H. Bauer, J.-L. Scharbarg, C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network", Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom., pp. 1-8, Sep. 2009.

[23] H. Bauer, J.-L. Scharbarg, C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach", IEEE Trans. Ind. Informat., vol. 6, no. 4, pp. 521-533, Nov. 2010.

[24] M. Li, G. Zhu, Y. Savaria, and M. Lauer, "Reliability enhancement of redundancy management in AFDX networks", IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2118–2129, Oct 2017.

[25] Ahmad Al Sheikh, Olivier Brun, Maxime Chéramy, Pierre-Emmanuel Hladik. Optimal Design of Virtual Links in AFDX Networks. 2012. [online] Available: https://hal.archives-ouvertes.fr/hal-00665755v1