A. DEMİRCİ

METU 2009

PERFORMANCE EVALUATION OF FLEXRAY NETWORKS FOR IN-VEHICLE COMMUNICATION

ALİ DEMİRCİ

NOVEMBER 2009

PERFORMANCE EVALUATION OF FLEXRAY NETWORKS FOR IN-VEHICLE COMMUNICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ DEMİRCİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2009

Approval of the thesis:

PERFORMANCE EVALUATION OF FLEXRAY NETWORKS FOR IN-VEHICLE COMMUNICATION

Submitted by **ALİ DEMİRCİ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Engineering** _____

Asst. Prof. Dr. Şenan Ece Schmidt
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Semih Bilgen
Electrical and Electronics Engineering Dept., METU _____

Asst. Prof. Dr. Şenan Ece Schmidt
Electrical and Electronics Engineering Dept., METU _____

Assoc. Prof. Dr. Özgür Barış Akan
Electrical and Electronics Engineering Dept., METU _____

Asst. Prof. Dr. Cüneyt Bazlamaçcı
Electrical and Electronics Engineering Dept., METU _____

Emrah Yürüklü, M.Sc.
TOFAŞ Turkish Automobile Factory A.Ş. _____

Date: 20.11.2009

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Ali DEMİRCİ

Signature :

ABSTRACT

PERFORMANCE EVALUATION OF FLEXRAY NETWORKS
FOR
IN-VEHICLE COMMUNICATION

Demirci, Ali
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Şenan Ece Schmidt

November 2009, 169 pages

The increasing use of electronic components in today's automobiles demands more powerful in-vehicle network communication protocols. FlexRay protocol, which is expected be the de-facto standard in the near future, is a deterministic, fault tolerant and fast protocol designed for in vehicle communication. In the near future, safety critical X-by-Wire applications will be available in the automobiles and FlexRay networks can be used to provide communication for the Electronic Control Units (ECUs) that perform related functions of X-by-Wire applications. In this thesis the performance of the FlexRay networks with various communication scenarios is evaluated in a real time environment and the results are presented. Communication scenarios investigate both static and dynamic segment of the FlexRay and allow evaluating the capabilities of the protocol. Several performance metrics such as utilization, static slot allocation, jitter are defined for the evaluation of the results.

Keywords: FlexRay, In-Vehicle Communication, X-by-Wire, Performance of the FlexRay networks.

ÖZ

FLEXRAY AĞLARININ ARAÇ İÇİ HABERLEŞMEDEKİ PERFORMANS DEĞERLENDİRMESİ.

Demirci, Ali
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Y. Doç. Dr. Şenan Ece Schmidt

Kasım 2009, 169 sayfa

Günümüz otomobillerinde artan elektronik birim kullanımı daha güçlü araba içi haberleşme protokollerine olan ihtiyacı doğurmaktadır. FlexRay protokolü ortaya çıkan bu ihtiyacı karşılayabilecek özelliklere sahip, gerekirci, hatalara dayanıklı ve hızlı bir haberleşme protokolüdür. Yakın bir gelecekte X-by-Wire fonksiyonları otomobillerde kullanılmaya başlanacaktır. FlexRay bu fonksiyonların işlemlerini yerine getirmek için kullanılacak Elektronik Kontrol Birimlerinin haberleşme ihtiyaçlarını karşılayacak yapıda bir protokoldür. Bu tez çalışmasında farklı haberleşme senaryolarında ve gerçek zamanlı bir donanım ortamında FlexRay ağlarının performansı incelenmiş ve sonuçlar değerlendirilmiştir. Bu haberleşme senaryoları protokolün hem statik hem de dinamik bölütünü kapsamaktadır. Sonuçların değerlendirilmesi içinde bazı performans metrikleri tanımlanmıştır.

Anahtar Kelimeler: FlexRay, Araç İçi Haberleşme, X-by-Wire, FlexRay Ağlarının Performansı

To My Grandfather

ACKNOWLEDGMENTS

I am in great debt to Asst. Prof. Dr. Şenan Ece Schmidt, my research advisor; she labored hard for two years to ensure I would accomplish my research goals.

I wish to thank ASELSAN A.Ş for giving me the opportunity of continuing my education.

I would like to thank TOFAŞ Türk Otomobil Fabrikaları A.Ş. for their support and encouragement to work on this costly and new technology.

I also thank Dr. Klaus Schmidt for his valuable contributions and comments during the life cycle of the research.

I would like to thank my parents and my wife for their patience and trust throughout my thesis.

I wish to thank to my friends and colleagues for their valuable support.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ v
ACKNOWLEDGMENTS vii
TABLE OF CONTENTS viii
LIST OF TABLES xi
LIST OF FIGURES xiv
CHAPTER
1. INTRODUCTION 1
2. IN-VEHICLE COMMUNICATION NETWORKS 5
2.1 Evolution of In-Vehicle Communication Networks 5
2.2 Car Domains 8
2.3 In-Vehicle Networks 12
2.3.1 Event Triggered Protocols 12
2.3.1.1 Controller Area Network (CAN) 12
2.3.2 Time Triggered Protocols 15
2.3.2.1 Local Interconnect Network (LIN) 15
2.3.2.2 Time Triggered Protocol (TTP) 16
2.3.2.3 Time Triggered Controller Area Network (TTCAN) 17
2.3.2.4 Byteflight 17
2.3.2.5 FlexRay 19
3. FLEXRAY PROTOCOL BASICS: SPECIFICATION, PERFORMANCE METRICS AND SCHEDULING 20
3.1 FlexRay Protocol Specifications 20
3.1.1 Media Access Control 20
3.1.1.1 Static Segment 21
3.1.1.2 Dynamic Segment 22
3.1.1.3 Symbol Window 24
3.1.1.4 Network Idle Time 25
3.1.2 FlexRay Frame Format 25
3.1.3 Composability of FlexRay 26
3.2 FlexRay Network Performance Metrics - Static Segment 27
3.2.1 Static Slot Allocation (SA) 27

3.2.2 Used Static Slot ID (U_{ID}) ... 29
3.2.3 Utilization (U) ... 30
3.2.4 Jitter (J) ... 31
3.2.5 Delay (D) ... 32
3.3 FlexRay Network Performance Metrics – Dynamic Segment ... 32
3.3.1 Delay (D) ... 32
3.3.2 Deadline Miss Ratio (DMR) ... 33
3.4 Message Scheduling for the FlexRay Network – Static Segment ... 33
4. DEVELOPMENT TOOLS USED FOR THE EVALUATION OF THE PROTOCOL ... 37
4.1 SK-91465X-100MPC Fujitsu FlexRay Evaluation Board ... 37
4.1.1 Softune Workbench Software Development Environment ... 38
4.1.2 Flash Programming Tool ... 39
4.1.3 FlexRay Communication Controller Driver ... 40
4.1.4 C-Compiler, Assembler and Linker ... 42
4.2 FlexConfig™ Developer – Universal FlexRay Configuration Tool ... 43
4.3 FlexCard Cyclone II SE ... 44
4.4 FlexAlyzer ... 44
5. DESIGN OF EXPERIMENTS ... 46
5.1 Work Flow in the Experiments ... 46
5.1.1 Examination of Message Set ... 46
5.1.2 Generation of Scheduling Scheme ... 47
5.1.3 Implementation of Configuration Parameters ... 57
5.1.4 Microcontroller Implementation ... 58
5.1.5 Analysis of the Results ... 58
6. EXPERIMENTS ... 59
6.1 Static Segment Experiments ... 61
6.1.1 Experiment 1: 3 Nodes, One Message per Static Slot ... 65
6.1.2 Experiment 2: 3 Nodes, Cycle Filtering without Jitter ... 69
6.1.3 Experiment 3: 3 Nodes, Cycle Multiplexing without Jitter ... 73
6.1.4 Experiment 4: 3 Nodes, Cycle Multiplexing with Jitter ... 76
6.1.5 Experiment 5: 3 Nodes, Cycle Multiplexing without Jitter – Signal Packing ... 79
6.1.6 Experiment 6: 6 Nodes, Cycle Multiplexing without Jitter ... 82
6.1.7 Experiment 7: 6 Nodes, Cycle Multiplexing with Jitter ... 84
6.1.8 Experiment 8 and 9: 6 Nodes, Cycle Multiplexing without Jitter – 2.5ms FlexRay Cycle ... 86
6.1.9 Experiment 10: 6 Nodes, Cycle Multiplexing without Jitter – Signal Packing ... 89
6.1.10 Summary and Discussion of the Results of the Static Segment Experiments ... 92

6.2 Dynamic Segment Experiments 99
6.2.1 Part 1 – Exploring the Values for Message Delays 101
6.2.2 Part 2 – Effect of Network Configuration Parameters 108
6.2.2.1 Varying Size of Dynamic Segment 110
6.2.2.2 Varying Message Payload Length 113
6.2.2.3 Varying Message Traffic Rate 115
6.2.2.4 Varying the Number of Messages 118
6.2.2.5 Determining Priorities with respect to Payload Lengths for Several Dynamic Segment Durations 120
6.2.3 Summary and Discussion of the Results of the Dynamic Segment Experiments 126
7. CONCLUSIONS 128
REFERENCES 132
APPENDIX A 135

LIST OF TABLES

TABLES

Table 5-1: Message Set Table 47
Table 5-2: Message Specific Configuration Parameters 53
Table 5-3: Cluster Configuration Parameters 53
Table 6-1: Fixed Configuration Parameters – Static Segment (Experiments 1, 2, 3, 4, 6, 7) 62
Table 6-2: Fixed Configuration Parameters – Static Segment (Experiments 5, 10) 62
Table 6-3: Fixed Configuration Parameters – Static Segment (Experiments 8, 9) 62
Table 6-4: Message set used in the Static Segment Experiments 64
Table 6-5: Experiment 1: Message Specific Configuration Parameters 65
Table 6-6: Experiment 2: Message Specific Configuration Parameters 70
Table 6-7: Experiment 3: Message Specific Configuration Parameters 74
Table 6-8: Experiment 4: Message Specific Configuration Parameters 77
Table 6-9: Experiment 5: Message Map and Configuration Parameters for the Network with 3 Nodes 80
Table 6-10: Experiment 6: Message specific parameters for new messages 82
Table 6-11: Experiment 7: Message Specific parameters for new messages 84
Table 6-12: Experiment 8 and 9: Message Specific Configuration Parameters 86
Table 6-13: Experiment 10: Message Map and Configuration Parameters for the Network with 3 Nodes 90
Table 6-14: Performance metrics calculated from the log data – Delay and Jitter 93
Table 6-15: Comparison of the Performance Metrics for the 3 Nodes Experiments – Static Segment 94
Table 6-16: Comparison of the Performance Metrics for the 6 Nodes Experiments – Static Segment 96
Table 6-17: Summary of the Static Segment Experiments 97
Table 6-18: Configuration parameters for dynamic segment – first part of the experiments 100
Table 6-19: Configuration parameters for dynamic segment – second part of the experiments 101
Table 6-20: Designed Experiments and Cluster Configuration Parameters – Part 1 102
Table 6-21: Configuration parameters that are kept constant 109
Table 6-22: Varying Size of Dynamic Segment Experiments– Configuration Parameters 110
Table 6-23: Varying message payload length experiments – Configuration Parameters 113
Table 6-24: Varying Message Traffic Load Experiments– Configuration Parameters 116
Table 6-25: Varying number of messages Experiments– Configuration Parameters 118

Table 6-26: Varying priorities with varying size of dynamic segment Experiments– Configuration Parameters 121
Table A-1: Static Segment, Experiment 1: Delay and Jitter values 135
Table A-2: Static Segment Experiment 2: Delay and Jitter values 136
Table A-3: Static Segment Experiment 3: Delay and Jitter values 136
Table A-4: Static Segment Experiment 4: Delay and Jitter values 137
Table A-5: Delay values for the case: 3 Nodes, 15 Messages, 8 bytes of payload, 472 Minislots, 53 Messages/sec 138
Table A-6: Delay values for the case: 3 Nodes, 15 Messages, 64 bytes of payload, 472 Minislots, 53 Messages/sec 139
Table A-7: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 472 Minislots, 53 Messages/sec 139
Table A-8: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 141 Minislots, 53 Messages/sec 140
Table A-9: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 141 Minislots, 300 Messages/sec 140
Table A-10: Static Segment Experiment 6: Delay and Jitter values 140
Table A-11: Static Segment Experiment 7: Delay and Jitter values 142
Table A-12: Delay Values for the case 6 Nodes, 16 Bytes of payload data, 472 minislots, 188 messages/sec 143
Table A-13: Delay Values for the case 6 Nodes, 36 Messages, 64 Bytes of payload data, 472 minislots, 188 messages/sec 143
Table A-14: Delay Values for the case 6 Nodes, 36 Messages, 254 Bytes of payload data, 472 minislots, 188 messages/sec 144
Table A-15: Delay Values for the case 6 Nodes, 36 Messages, 254 bytes payload, 141 minislots, 188 messages/sec 145
Table A-16: Delay Values for the case 6 Nodes, 36 Messages, 254 bytes payload, 141 minislots 719 messages/sec 146
Table A-17: Varying Dynamic Segment Length – 50 minislots 147
Table A-18: Varying Dynamic Segment Length – 100 minislots 147
Table A-19: Varying Dynamic Segment Length – 200 minislots 148
Table A-20: Varying Dynamic Segment Length – 307 minislots 149
Table A-21: Varying Payload Length – 16 Byte 150
Table A-22: Varying Payload Length – 64 Byte 151
Table A-23: Varying Payload Length – 128 Byte 151
Table A-24: Varying Payload Length – 254 Byte 152
Table A-25: Varying Message Traffic Rate – 480 messages/second 153

Table A-26: Varying Message Traffic Rate – 238 messages/second 154
Table A-27: Varying Message Traffic Rate – 127 messages/second 155
Table A-28: Varying Number of Messages – 36 messages (6 per node) 155
Table A-29: Varying Number of Messages – 48 messages (8 per node) 156
Table A-30: Varying Number of Messages – 66 messages (11 per node) 157
Table A-31: Varying Number of Messages – 84 messages (14 per node) 159
Table A-32: Decreasing Payload Length – dynamic segment length of 100 minislots 160
Table A-33: Decreasing Payload Length – dynamic segment length of 200 minislots 161
Table A-34: Decreasing Payload Length – dynamic segment length of 307 minislots 162
Table A-35: Increasing Payload Length – dynamic segment length of 100 minislots 163
Table A-36: Increasing Payload Length – dynamic segment length of 200 minislots 163
Table A-37: Increasing Payload Length – dynamic segment length of 307 minislots 164
Table A-38: Static Segment Experiment 5, 3 Nodes, Signal Packing 165
Table A-39: Static Segment Experiment 8, 6 Nodes, 2.5ms Cycle Duration, Scheduling Offsets are synchronized with generation offsets 166
Table A-40: Static Segment Experiment 9, 6 Nodes, 2.5ms Cycle Duration, First occurrence of the messages are at the same cycle 167
Table A-41: Static Segment Experiment 10, 6 Nodes, Signal Packing 168

LIST OF FIGURES

FIGURES

Figure 2-1: In-Vehicle Networks [3] 6
Figure 2-2: In-Vehicle Networks for functional domains at BMW [23] 9
Figure 2-3: Standard CAN 2.0A Data Frame Format 14
Figure 3-1: FlexRay Communication Cycle Timing Hierarchy [13] 20
Figure 3-2: Static Segment Message Transmission on Redundant Channels [13] 22
Figure 3-3: Frame Transmission in Dynamic Segment [13] 23
Figure 3-4: FlexRay Frame Format [13] 25
Figure 3-5: Allocation of the static slot to the message m over the cycles 0 to 63 34
Figure 3-6: Full Allocation of a static slot for 4 messages by using Cycle Multiplexing 35
Figure 3-7: Partial Allocation of a static slot for 2 messages by using Cycle Multiplexing 36
Figure 4-1: SK-91465X-100MPC Fujitsu FlexRay Evaluation Board 38
Figure 4-2: Softune Workbench user interface 39
Figure 4-3: Flash Programmer user interface 40
Figure 4-4: Architecture of the FlexRay Driver 41
Figure 4-5: Program flow of the FlexRay Driver 42
Figure 4-6: FlexConfig user interface 43
Figure 4-7: FlexCard Cyclone II SE 44
Figure 4-8: FlexAlyzer User interface 45
Figure 5-1: Work Flow for the Experiments 46
Figure 5-2: Static Segment Priorities 49
Figure 5-3: Scheduling map of a static slot for the case, allocation with minimum jitter 51
Figure 5-4: Scheduling map of a static slot for the case, minimum *SA* (Jitter is allowed) 52
Figure 5-5: Physical FlexRay Frame [22] 55
Figure 5-6: Configuration of Static Segment 57
Figure 6-1: 3 Node Network Setup 60
Figure 6-2: 6 Node Network 60
Figure 6-3: Experiment 1, Static slot allocation, one slot is allocated for each message in every cycle 67
Figure 6-4: Experiment 1 – delay (µsec) & jitter (µsec) vs. Message ID 68
Figure 6-5: Experiment 2, Static slot allocation, cycle filtering is used 70
Figure 6-6: Experiment 2 – delay (µsec) & jitter (µsec) vs. Message ID 73

Figure 6-7: Experiment 3, Static slot allocation, message filtering and cycle multiplexing is used ... 74
Figure 6-8: Experiment 3 – delay (µsec) & jitter (µsec) vs. Message ID 76
Figure 6-9: Experiment 4 – delay (µsec) & jitter (µsec) vs. Message ID 79
Figure 6-10: Experiment 5 – delay (µsec) & jitter (µsec) vs. Message ID..... 81
Figure 6-11: Experiment 6 – delay (µsec) & jitter (µsec) vs. Message ID..... 83
Figure 6-12: Experiment 7 – delay (µsec) & jitter (µsec) vs. Message ID..... 85
Figure 6-13: Experiment 8 – delay (µsec) & jitter (µsec) vs. Message ID (message generations are synchronized with scheduling offsets) 88
Figure 6-14: Experiment 9 – delay (µsec) & jitter (µsec) vs. Message ID (first occurrence of the messages that belongs to a specific node happens in the same cycle)..... 89
Figure 6-15: Experiment 10 – delay (µsec) & jitter (µsec) vs. Message ID..... 92
Figure 6-16: Maximum (µsec) & Average (µsec) Delays vs. Priority Levels: 15 Messages, 472 Minislots, - 8, 64 and 154 Bytes of Payload Data 103
Figure 6-17: Maximum (µsec) & Average (µsec) Delays vs. Priority: 15 Messages, 254 Bytes of Payload Data - Dynamic segment length of 472 and 141 minislots..... 104
Figure 6-18: Maximum (µsec) & Average Delays (µsec) vs. Priority Level: 15 Messages, 254 Bytes of Payload Data, Dynamic segment length of 141 minislots - traffic rates of 53 messages/sec and 300 messages/sec 105
Figure 6-19: Maximum & Average Delays (µsec) vs. Priority Levels: 36 Messages, Dynamic Segment length of 472 minislots, 188 messages/sec – 16, 64 and 254 Bytes of payload data,..... 106
Figure 6-20: Maximum & Average Delays (µsec) vs. Priority: 36 Messages, 254 Bytes of payload data, 188 messages/sec - Dynamic Segment length of 472 and 141 minislots,..... 107
Figure 6-21: Maximum & Average Delays (µsec) vs. Priority - 6 Nodes, 254 Bytes of payload data, Dynamic Segment length of 141 minislots, 719 messages/sec and 188 messages/sec message traffic rate 108
Figure 6-22: Maximum delay (µsec) vs. Priority: the case of varying dynamic segment length..... 111
Figure 6-23: Average Delay (µsec) vs. Priority: the case of varying dynamic segment length 112
Figure 6-24: Standard Deviation (µsec) vs. Priority: the case of varying dynamic segment length . 112
Figure 6-25: Maximum Delay (µsec) vs. Priority: the case of varying payload length 114
Figure 6-26: Average Delay (µsec) vs. Priority: the case of varying payload length 114
Figure 6-27: Standard Deviation (µsec) vs. Priority: the case of varying payload length..... 115
Figure 6-28: Maximum delay (µsec) vs. Priority: the case of varying message traffic load..... 116
Figure 6-29: Average Delay (µsec) vs. Priority: the case of varying message traffic load..... 117
Figure 6-30: Standard Deviation (µsec) vs. Priority: the case of varying message traffic load 117
Figure 6-31: Maximum Delay (µsec) vs. Priority: the case of varying number of messages 119
Figure 6-32: Average Delay (µsec) vs. Priority: the case of varying number of messages..... 119
Figure 6-33: Standard Deviation (µsec) vs. Priority: the case of varying number of messages..... 120

Figure 6-34: Scheduling Scheme – 16 Bytes messages have the highest priority..... 121
Figure 6-35: Scheduling Scheme – 254 Bytes messages have the highest priority..... 122
Figure 6-36: Maximum Delay (μsec) vs. Priority: the case of 100 minislots of Dynamic segment . 123
Figure 6-37: Maximum Delay (μsec) vs. Priority: the case of 200 minislots of Dynamic segment . 124
Figure 6-38: Maximum Delay (μsec) vs. Priority: the case of 307 minislots of Dynamic segment . 124
Figure 6-39: Average Delay & Standard Deviation (μsec) vs. Priority: the case of 100 minislots of Dynamic Segment..... 125
Figure 6-40: Average Delay & Standard Deviation (μsec) vs. Priority: the case of 200 minislots of Dynamic Segment..... 125
Figure 6-41: Average Delay & Standard Deviation (μsec) vs. Priority: the case of 307 minislots of Dynamic Segment..... 126

CHAPTER 1

INTRODUCTION

The number of electronic systems that are used in the automobiles is increasing and mechanic, hydraulic systems are being replaced by new electronic systems which are composed of Electronic Control Units (ECU). These systems are exchanging large amount of data among themselves. This data consist of signals that are encoded into messages. 2500 different signals can be exchanged among the ECUs of an automotive system having several subsystems [8]. Furthermore these messages have different characteristics. There are mainly two types of messages defined as periodic and sporadic. Periodic messages are generated with fixed time periods and their generation times are known deterministically. On the other hand sporadic messages are generated in response to the occurrence of events, and their timing properties are not known prior to the run time. In this thesis we use the terms signals and messages interchangeably.

In the early days of the in-vehicle electronic systems, the signal exchange was realized by point-to-point communication. As the number of messages increased this approach was replaced by communication networks to decrease the wire harness and cost as well as increase the maintainability. In-vehicle communication networks are divided into two groups according to their timing characteristics as event-triggered and time triggered networks. The type of the network to be used is decided according to the requirements of the automotive applications and the type of messages that they are needed to carry. Controller Area Network (CAN) is a famous example for event triggered networks. For time triggered networks Local Interconnect Network (LIN), Time Triggered Protocol (TTP), Time Triggered

CAN (TTCAN), Byteflight, and FlexRay can be given as the most important examples. Each network has advantages and disadvantages over others considering the cost, complexity, and performance. Apart from these examples several automobile manufacturers use their own developed networks.

Today CAN is the most widely used network protocol because of its low cost, robustness, and bounded communication delay. Its data rate can go up to 1 Mbps. X-by-Wire systems (such as brake-by-wire, steer-by-wire, and drive-by-wire) will be introduced in the cars in the near future, and the communication requirements of these systems are much more demanding than today's systems. These requirements are the real-time transmission guarantees, scalability of the network to include new nodes as required, high level of safety and fault tolerance as well as 5 Mbps of network bandwidth [5]. 1 Mbps bandwidth of the CAN has become insufficient for X-by-Wire systems. Furthermore Event-Triggered structure of the CAN is not appropriate to satisfy the timing and safety requirements for the message transmissions. For these reasons a new networking protocol is required for the automobile manufacturers to replace CAN.

Three candidates are developed to be used with the future X-by-Wire systems by various Development Groups. These candidates are TTCAN which is developed by Automotive Systems Research Group (ASRG), TTP which is developed by Brite Euram Project "X-by-Wire" and ESPRIT OMI Project "Time Triggered Application - TTA" at the Technical University of Vienna , and FlexRay developed by FlexRay Consortium. TTCAN is using the available CAN structure with an application layer that satisfies the time-triggered communication. However the limitations of the CAN such as low bandwidth and lack of fault tolerance makes it insufficient for the requirements of X-by-Wire systems. TTP fulfills the requirements and gives enough safety level and bandwidth. However because of its high cost and inflexible structure, it could not find support from the automobile manufacturers. The high-bandwidth that is flexibly allocated among the messages via time slots, support for both periodic and sporadic messages and the fault tolerance properties of the FlexRay protocol makes it the best candidate for X-by-

Wire applications and it is likely to become the de-facto standard in the near future [21]. Furthermore the first use of FlexRay technology for adaptive drive suspension system data exchange in the BMW X5 Sports Activity Vehicle (SAV) is presented by BMW in 2007 and later in 2008 BMW X6 was introduced with fully utilized FlexRay network [16].

In this thesis we investigate the performance of the FlexRay network by using FlexRay nodes implemented in hardware. We define performance metrics that measure how effectively the bandwidth is used (*utilization*, *used static slot IDs*, *static slot allocation*) as well as the achieved quality of service by the messages (*deadline misses, delay*, *jitter).* Similar to other in-vehicle networks, the FlexRay network is configured and the transmission schedule of the messages is determined before the network starts to operate. We evaluate the real time performance of the FlexRay protocol under different scheduling and network configurations with different traffic loads according to our defined performance metrics. Several experiments are designed to evaluate the transmission of both periodic and sporadic messages. The purpose, analysis and results of each experiment are presented in the related sections.

We observe that by applying appropriate message schedules and choosing the right configuration parameters the performance of the network can be improved. We also demonstrate the trade offs between the metrics such as utilization and delay, the amount of allocated slots and jitter. Furthermore the effects of several network configurations on the performance metrics are explored. Advantages and disadvantages of each configuration are investigated.

The remainder of the thesis is organized as follows: in Chapter 2 we review the evolution of in-vehicle communication networks and the functional car domains and their requirements; then properties of selected network protocols are presented. Then in Chapter 2 FlexRay protocol details and defined performance metrics are explained. Then the message scheduling methods that are applied in the experiments are given. Development tools used in the experiments are presented in

Chapter 4, and in Chapter 5 the process followed for the experiments are given. In Chapter 6 designed experiments, their analysis and evaluation of the results are presented. Chapter 7 concludes the thesis work and presents suggestions for future work.

CHAPTER 2

IN-VEHICLE COMMUNICATION NETWORKS

Today's automotive systems are complex distributed systems with several in-vehicle networks and the demands for the capabilities of these networks are increasing. In this section, the progression of the in-vehicle networks from pure mechanical systems to today's cars is given at first. Then the car domain is explained to understand the requirements of the communication networks which are needed to be used in different domains of the cars. Finally the most popular in-vehicle networks in use or ready to use in the automobile systems are presented.

2.1 Evolution of In-Vehicle Communication Networks

Starting from 1970s purely mechanical or hydraulic systems are replaced with safe and efficient electronic systems. Furthermore new features are constantly added to automobiles in parallel to the technology improvements. These electronic systems are becoming more reliable and also their maintainability increase and cost reduce by the help of software technologies. At the beginning each new function or system was realized with a stand alone Electronic Control Unit (ECU). The data exchange among ECUs, sensors and actuators was realized by point to point connections. However this method became insufficient as the number of ECUs increased, because every new ECU meant more cables, connectors, weight, power demand and cost. This reduced the performance, reliability and maintainability of the whole system. For this reason automotive industry started to use networks for data exchange among ECUs. The computer data networks were the starting point for in-Vehicle network development. However, instead of only the data transfer, the applications were driven by control strategies which required real time, safe and

robust operation. Every company was developing its own network structure therefore there was little interest for common network standards. However as the standardized networking concept became more important, external suppliers' role in the automotive industry became more important. Cost of the integration of new technologies was reduced and mature standardized protocols came into place [1] [25].

In the early 1980s Bosch developed an automotive bus protocol, Controller Area Network (CAN), and it was first used in Mercedes production cars in 1990s. Later it became an ISO standard in 1994. Now CAN is the most widely used network in automotive control systems due to its low cost, robustness, and bounded communication delay. Also additionally several other communications networks are used in different parts of the vehicles as smaller sub networks within the vehicle as shown in Figure 2-1 [2].

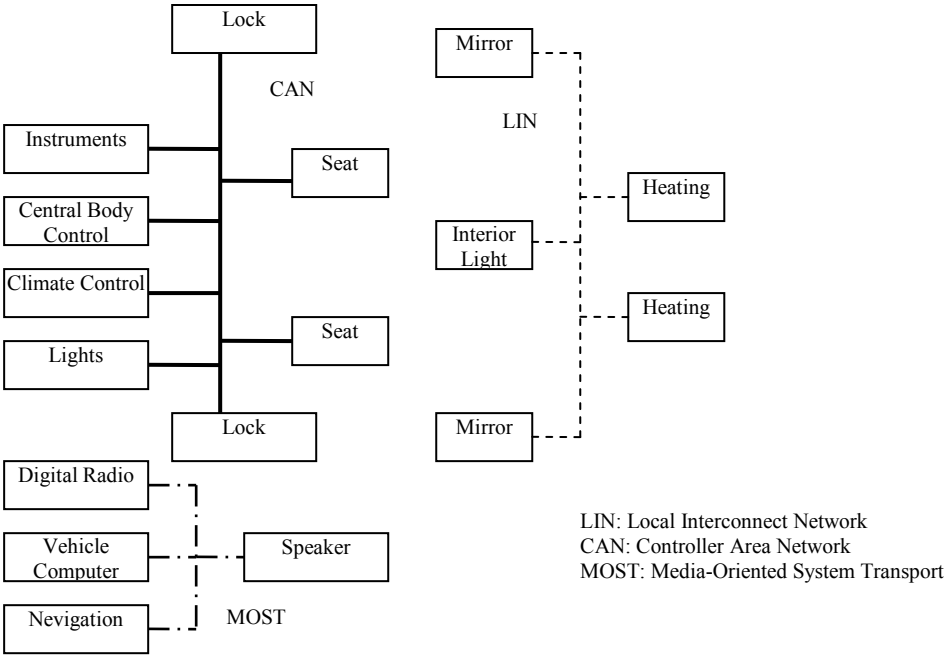

Figure 2-1: In-Vehicle Networks [3]

The automotive manufacturers have to continue their improvements to respond to customer demands that have more features with a low price. As the number of features increases the number of electronic systems that realize these new features also has to increase.

An important direction for the vehicles of the future is the *X-by-Wire* technology which is gradually replacing the mechanical and hydraulic systems by electronic ones to provide safer and more comfortable driving experience to the drivers as well as more efficient operation. An example for X-by wire applications is "throttle-by wire" which came into use in 1990s and was employed in a number of cars including BMW 7 and Chevrolet Corvette. Today it exists in a large number of car models. In the conventional approach the air intake for the combustion process is regulated by the throttle plate. The accelerator pedal is connected to the throttle plate via a mechanical linkage. In throttle by wire the throttle plate is controlled by a motor that is controlled by a microcontroller that determined the correct throttle position. When the driver steps on the accelerator pedal a sensor on the pedal transmits the pressure to the microcontroller which computes the correct throttle position. This information is transmitted to the motor that controls the throttle plate. Achieving the optimal throttle position at all times cannot be achieved by human drivers and it improves the fuel economy of the car and the emissions. An interesting fact to note is that many drivers are not aware of this technology and do not know that the accelerator pedal does not control the throttle plate anymore. Starting from applications such as throttle by wire that impacts the vehicle's performance, the new goals are steer-by-wire and brake by wire which also impacts the vehicle's safety [27]. X-by-Wire technologies require fast message transmission with delay and bandwidth guarantees as well as, reliability and safety.

Control messages are continuously exchanged between several ECUs in cars. The properties of these messages have an important effect on the network topology. By considering the timing behaviors and transmission constraints, messages can be classified into two main categories, *periodic* and *sporadic* messages. Between two consecutive messages there is a definite time difference for periodic messages. For

this reason the time instants that the messages generated are known prior to the transmission. Jitter is an important parameter for periodic messages. From the control algorithms point of view the best performance can be achieved without any jitter. The sporadic messages are generated in response to the occurrence of an event and the time difference between two consecutive messages is not definite. However the minimum time between message occurrences can be estimated from the physical and natural constraints. Opening the windows or doors of the car or pressing on the brakes are examples of this kind of events. The time period between message generation by sender ECU and reception by receiver ECU is defined as *delay*. The delays of the real time messages are restricted to be less then a maximum value called as *deadline.* The network bandwidth and transmission protocol are main parameters that controls the delay.

2.2 Car Domains

In vehicles there are several function domains, with different functionalities and requirements. While some of them need high safety; timely delivery is the most important aspect for the other ones. For this reason in today's cars there are several networks serving to different parts of the electronic systems as shown in Figure 2-2. Typical functional domains according to their features and constraints can be classified into 5 categories. These categories are power train, chassis & suspension, body electronics, infotainment and active and passive safety. Note that FlexRay is used as the backbone that connects each network to others because of its high capacity.

Power train is the collection of functions that are responsible for the control of engine. Since the speed of the engine is very high, to control the engine complex control functions and very short periods for the message transmissions are needed. For this reason high processing power for the controllers and multitasking with fast delivery of messages is important. Strict time constraints for the messages have to be satisfied. Furthermore power train functions perform frequent data exchange with other domains.

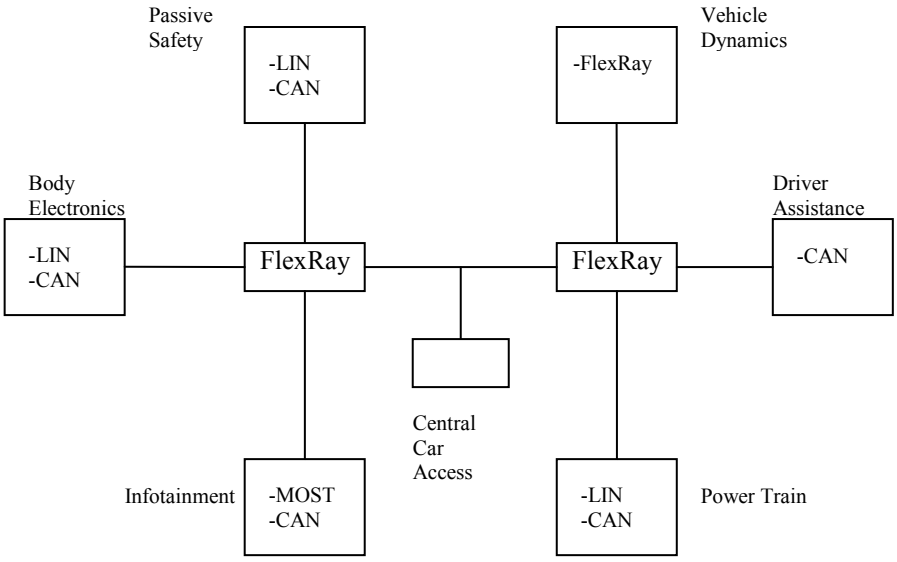

Figure 2-2: In-Vehicle Networks for functional domains at BMW [23]

The systems that control the interaction of the car with the road and chassis components (wheel, suspension) are the building blocks of *chassis domain*. The main inputs to these systems are the driver's actions and environmental conditions such as wind, temperature, road profile. ABS (Antilock Braking System), ESP (Electronic Stability Program), ASC (Automatic Stability Control) and 4WD (4 Wheel Drive) are some examples of functions that belong to chassis domain. Communication requirements of this domain are very similar to Power Train. Since these functions are directly related to vehicle's stability and dynamics, safety is the most important requirement for the chassis domain. Furthermore x-by-wire applications such as steer-by-wire, brake-by-wire, will be implemented in the cars in the near future. The electronic systems employed in these applications will be very intelligent devices, networks, and software components that implement filtering, control, and diagnosis functionalities. This increases the safety requirements of the chassis domain. Apart from the safety, x-by-wire applications demand for much more bandwidth from present chassis domain functions. Power train and chassis functions operate mainly as closed-loop control systems and their implementation is moving towards a time triggered approach.

Body electronics such as wipers, doors, windows, seats, lights, mirrors and climate control are controlled almost entirely with software based systems. They do not have strict performance constraints however they are subject to so many communications among them and this leads to a complex distributed system architecture. Since these nodes do not require large bandwidth and do not have strict timing requirements, low cost networks, such as LIN (Local Interconnect Network), are used for message exchange. The body domain integrates a central subsystem called "central body electronics" which is implemented with a higher bandwidth network such as CAN and serves as a backbone for other low-cost networks. High computation power, fault tolerance and reliability properties are the requirements of the body electronics. The body functions are activated by the driver or passengers, so that event triggered networks are suitable for this domain.

Telematic functions are becoming more important for the customers and also new functions are coming into vehicle domain. Some examples of Telematic functions are car radio, CD, DVD, navigation systems, hands-free phones. They enable communication with systems inside the vehicle and also information exchange with the outside world. These functions need high network bandwidth because a very large amount of data exchange is required within the vehicle and with outside world. Since outside world is also a node for the system, wireless communication has to be supported. Different than other domains strict deadline constraints for multimedia messages, bandwidth sharing and information security are very important for these applications.

The domain of *active and passive safety* includes the electronic based systems to ensure the safety of driver and passengers. Impact and rollover sensors, deployment of airbags and belt pretensioners, Tire Pressure Monitoring and Adaptive Cruise Control (ACC) are some examples of this domain [1].

The performance requirements such as maximum delay, fault tolerance, reliability of the networks to be used in these vehicle domains are different. Performance, cost, safety, and bandwidth requirements lead to different kinds of networks used

in different domains of vehicles. In 1994, the Society of Automotive Engineers (SAE) classified the vehicle networks into 4 categories based on their bit transfer rates. Specific standards exist for Class A, Class B and Class C networks. For Class D networks SAE has not defined a standard yet.

Class A networks:

Class A networks have a data rate of less than 10 Kbit/s. Their costs are low compared to others and mainly used for transmitting simple control messages. Considering body electronic requirements, this kind of networks are well suitable for this applications. Examples of class A networks are Local Interconnect Network (LIN) and Time Triggered Protocol/A (TTP/A).

Class B networks:

Networks with a data rate between 10 and 125 Kbit/s are classified as Class B. They are used for general information transfer such as instruments and power window. Low-Speed CAN and J1850 belong to this class.

Class C networks:

Class C networks have data rate between 125 and 1000 Kbit/s. Functions that need high bandwidth require this class of networks. High-Speed CAN is the main representative of this class and used mainly in power train and chassis applications.

Class D networks:

Networks with a data rate higher than 1 Mbit/s belongs to this class. Since telematic applications need high bandwidth and low message latency, they use Class D networks such as Media Oriented Systems Transport (MOST). The future x-by-wire applications also need high bandwidth and different then Telematics domain fault tolerance and reliability are additional constraints. TTP/C and FlexRay are suitable for x-by-wire applications. However automotive industry is adopting FlexRay as the de-facto standard for use in new development projects.

In today's vehicles one can see all these kind of networks together, integrated by gateways. For example the Volvo XC90 contains up to 40 ECUs interconnected by a LIN bus, a MOST bus, a low-speed CAN, and a high-speed CAN [1].

2.3 In-Vehicle Networks

Transmission protocols are classified into two main categories according to their bus access control properties. In *Time-Triggered networks* Time Division Multiple Access (TDMA) based medium access control mechanisms are used. Time is divided into equal parts which are called *cycles* and these identical cycles are repeated continuously. Cycles are also divided into *slots* where individual transmissions take place. Prior to the transmission each ECU knows which slot it has permission to send. If the ECU, that possesses the particular slot, does not send any message no transmission takes place in that slot and cycle. This may lead to bandwidth loss. However, since scheduling is done prior to run time, delay is constant and predictable. Furthermore these networks can be realized with simple bus access algorithms. Time-triggered networks are highly suitable for periodic message transmission. In the case of *Event-Triggered networks,* messages are transmitted in response to occurrence of events. Opposed to time-triggered networks, there is no bandwidth allocation for ECUs. This leads to a flexible and efficient network bandwidth usage. High network utilization can be achieved. However in this case timing behavior of the messages is not predictable, and also delays are not constant. Furthermore complex bus access protocols are needed for fair and efficient usage of bandwidth. Event-triggered networks are suitable for sporadic messages [1] [3] [25].

2.3.1 Event Triggered Protocols

2.3.1.1 Controller Area Network (CAN)

A Controller Area Network (CAN) is an asynchronous serial bus network that connects devices, sensors and actuators in a system or subsystem for control applications. It was developed by Bosch GmbH in the middle of 1980s and later in 1993 it has been adopted as an international standard by International Organization

for Standardization (ISO) as ISO11898-1. Today it is the most commonly used automobile communication protocol around the world. More than 100 million CAN ECUs was sold in 2000. Although its data rate can go up to 1 Mbps, because of high EMC shielding cost for data rates exceeding 500 kbps, the higher speeds are not effective for automobile applications. Multiple CAN networks with different transmission rates are used in different domains of the vehicles. For example while in telematics domain a low-speed CAN network with 125 kbps data rate is used, higher speed CAN with 500 kbps can be used in power train functional domain. The choice depends upon the requirements of that domain. Low cost and low power consumption properties of low speed CAN and its help to reduce the wire harness makes it very suitable for telematics. On the other hand higher bandwidth requirement of power train domain leads to use of high speed CAN [2] [3] [6].

As an event triggered message transmission network, CAN uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) for the Medium Access Control (MAC) protocol. In addition to the CSMA/CD CAN also includes various methods for error detection and handling. It uses nonreturn-to-zero (NRZ) bit representation with a bit stuffing of five. If the network is idle any ECU can start a transmission. If there is a collision, the ECU with lower priority stops transmission to leave the network free for the higher priority message. Collision detection is realized by bitwise arbitration using the identifier of each message. The ECU that sends a recessive bit, but detects a dominant bit concludes that another ECU with higher priority also wants to send a message and then stops. Since the dominant bit is logical zero, the messages with smaller identification number has higher priority [4].

CAN Message Format:

CAN has four different frame types, Data Frame, Remote Frame, Error Frame and Overload Frame. A data frame is composed of seven different bit fields. Standard CAN 2.0A Data Frame Format is shown in Figure 2-3. Data frame begins with Start of Frame (SOF) bit. It is followed by eleven bits Frame Identifier and the

Remote Transmission Request (RTR) bit. Identifier and RTR bit form the arbitration field. The Control Field consists of six bits, four bits to indicate data length and two bits reserved for future expansions. Application data has variable length according to the message and can be zero to eight bytes. CRC field follows the application data and it enables the receiver to check if there is error in the incoming message sequence. Acknowledgement (ACK) field is used by transmitter to see if any receiver gets a valid frame. The message frame ends with End of Frame. In the extended format (CAN 2.0B) arbitration field consist of 29 bit identifier [1].

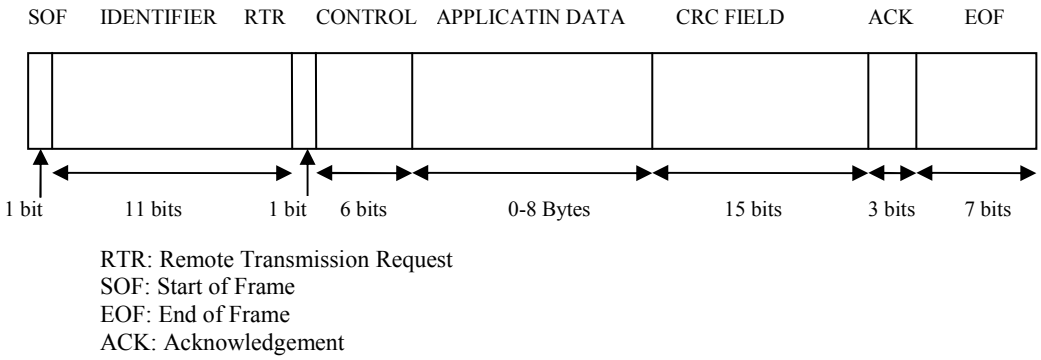

Figure 2-3: Standard CAN 2.0A Data Frame Format

Considering the nondestructive arbitration process, CAN is able to resolve collisions in a deterministic way. By this way message delays can be considered as bounded. However if nodes are allowed to produce asynchronous messages on their own, it is not possible to know when the message will be sent. This is because one can not now how many times a message will experience collisions. This behavior leads to unexpected jitter and some messages may miss their deadlines. Varying jitter can decrease the performance of control algorithms. For safety critical applications this may lead to dangerous results. This conclusion is valid for almost all event triggered protocols [5].

Another drawback of CAN, when it is used in safety critical applications, is the problem of *babbling idiot*. That means, one of the ECUs can repeatedly transmit

high priority messages because of an internal error and block the whole network. Since this faulty transmission is not the result of physical fault, it can not be detected by CAN [5].

Because of its limited data rate, CAN also suffers from the performance limitations. The maximum allowed data rate is 1 Mbps and over 500 kbps is very costly. Furthermore this limitation is because of physical media and can not be solved easily. In the near future automotive applications such as X-by-Wire systems will demand much more bandwidth than a CAN network can supply. In particular several car companies consider a net data rate of at least 5 Mbps is needed for X-by-Wire applications. Moreover the medium access technique adopted by the network has to be somehow scalable, that is it should be possible to increase the bit rate as it is required by the control applications [5].

2.3.2 Time Triggered Protocols

2.3.2.1 Local Interconnect Network (LIN)

LIN is a cost-competitive, time triggered, serial communication system designed for localized vehicle electrical networks. It is an inexpensive network based on common UART/SCI interface hardware, providing network speeds up to 20 kbps on a single wire. It was initiated in 1998 by a consortium of automotive companies together with Motorola and was standardized in 2000 (LIN 1.1) and 2003 (LIN 2.0). It was first introduced in production car series in 2001. As SAE Class A network, LIN is typically used in body electronics and comfort subsystems to control devices such as seat control, light sensors, climate control, rain sensors, sunroofs and doors. These subsystems are later interconnected using relatively higher speed networks. Commonly CAN is used as high speed network, with a gateway. It is suitable for non safety related subsystems [7] [8].

A LIN cluster consists of one “master” ECU and several (up to 16) “slave” ECUs connected to a common bus. The master is typically a moderately powerful microcontroller, whereas the slaves can be less powerful and cheaper microcontrollers. Maximum of eight byte of data can be sent in a message. The

master decides when and which frame will be transmitted according to the schedule table, so there is no need for arbitration. Since it is time triggered, message delays are predictable.

2.3.2.2 Time Triggered Protocol (TTP)

There are two versions defined for TTP as TTP/C and TTP/A. TTP/C is a deterministic protocol intended for SAE Class C applications. It was developed by the Brite Euram Project "X-by-Wire" and ESPRIT OMI Project "Time Triggered Application - TTA" at the Technical University of Vienna [4]. The TTP integrates all the services necessary for fault tolerant real time systems, providing network speeds up to 25 Mbps on two replicated channels. Bus access is realized with TDMA and each ECU connected to the network can send message in a predetermined TDMA slot. This ensures predicted message delay for delivery just like other time triggered protocols. TTP/C implements several fault tolerant mechanisms such as atomic broadcast using membership service, distributed clock synchronization and bus guardians. The known "babbling idiot" problem is solved with bus guardians. At a specific time instant, the network can be seen as peer-to-peer that means only two ECUs, transmitter and receiver, can use the bus. By this way single point failures do not affect the whole network, but related ECU. A message frame can carry data up to 240 bytes. Each ECU keeps its own scheduling information in a Message Descriptor List (MEDL). Furthermore MEDL also includes all scheduling information of the cluster. Scheduling is done statically prior to the run time. When the schedule of an ECU is changed all the remaining ECUs' MEDL table also has to be changed [1] [9] [14].

There are two types of frames in TTP/C protocol. These are Initialization Frames (I-Frames) and Normal Frames (N-Frames). I-Frames are used for initialization purpose and also for the membership service. An ECU can be included to membership service and start communication after receiving an I-Frame. N-Frames used in normal operation and contains the data. TTP/C frames do not include destination addresses because every ECU knows when it will send or receive data

by MEDL table. This structure decreases the framing over head and also increases the channel utilization [11].

Because of its limited flexibility and high cost TTP/C might not be used for X-by-Wire systems by automotive industry in the future.

TTP/A is a lower cost version of TTP/C, for SAE Class A applications. This version is also TDMA based and has master/slave architecture like LIN.

2.3.2.3 Time Triggered Controller Area Network (TTCAN)

TTCAN was introduced in 1999 as a time triggered session layer on top of the existing data link and physical layers of CAN. It is standardized by ISO and intended for X-by-Wire applications. It allows CAN to be used for time triggered messages, so increasing determinism, reliability, synchronization. Since it is on top of CAN, transition from CAN to TTCAN will be easy for car manufacturers [3].

In TTCAN a specific node, called time master, transmits a reference message indicating the start of a time cycle. A time cycle is divided into a number of slots each of which can be assigned statically to a specific node for deterministic transmission or group of nodes that compete for it by CAN arbitration for event triggered messages. If a time master fails, another potential node, any of other nodes, may be the time master [4].

However TTCAN does not provide the same level of fault tolerance as TTP and FlexRay, which are other two candidates for future X-by-Wire systems.

2.3.2.4 Byteflight

Byteflight is a high speed communication network developed for safety critical applications. It has net data rate of 5 Mbps and gross data rate of 10 Mbps. Optical fiber is used in physical layer for 10 Mbps data rate to avoid electromechanical interference problems. Protocol is structured in a way that it has the advantages of both time-triggered and event-triggered communication schemes. A flexible TDMA type medium access control scheme is used for transmission scheduling.

Transmission of data is organized in cycles, and each cycle starts with a Synchronization Pulse (SYNC). SYNC is sent by a node called as SYNC master. Any node in the network can be a SYNC master. These pulses make the time base common for all ECUs. The time interval between SYNC pulses is 250 microseconds. The protocol combines time and priority controlled bus access. Every message in the cluster has a unique identifier (ID) based on its priority. High priority messages have low IDs. The time between two SYNCs is divided into slots and every ECU keeps a slot counter. Value of this counter can be at most 255. Following a SYNC, ECUs start incrementing their counters with a predefined rate. When the counter value equals to the ECUs message ID, it transmits related message. While message transmission takes place all the ECUs pauses their counters. When transmission is completed counters return to incrementing their counters from where they left. Counters are reset when they reach 255 or a SYNC is transmitted. By this way no messages miss their transmission slot. This guarantees deterministic latencies for specific number of high priority messages. The number depends on the length of messages.

Byteflight serves as a highly flexible network protocol. FTDMA scheme allows efficient use of bandwidth especially for low priority messages. When no transmission takes place only a small portion of time is wasted. This time is shorter then any message duration. Deterministic behavior of high priority messages makes Byteflight suitable for safety critical applications. One message is allowed to be sent once in one cycle. This prevents the network from babbling idiot problem. However in case of large number of high priority messages, low priority ones may not use the bus. This may lead to message missing failures. Hence Flexible TDMA is not as good as time triggered schemes from the point of view of determinism. Another drawback of the Byteflight is that correct network operation depends on one SYNC master. In case of a failure of SYNC master, whole network can be malfunctioned. This can be prevented by one or more backup masters [4] [5] [15].

2.3.2.5 FlexRay

FlexRay consortium was founded by BMW, DaimlerChrysler, Motorola and Philips Semiconductor in 2000, and the first version of the Protocol Specifications (V.2.0) was released in 2004. The current version of the specifications is V.2.1 which was released in 2005. FlexRay is a high data rate, fault tolerant network protocol designed for the needs for future embedded in-vehicle electronic systems. Current trend shows that FlexRay will be de-facto standard for the future in-vehicle networking applications, and CAN networks will be replaced by FlexRay networks [3] [13].

FlexRay protocol combines both time-triggered and event-triggered message transmission. While time triggered part is realized by TDMA protocol, a flexible TDMA protocol is used for event triggered part. FlexRay has a scalable data rate options as 2.5, 5 and 10 Mbps that is suitable for X-by-Wire applications. To increase the fault tolerance it supports two redundant communication channels, which increases the reliability of the protocol that is required by x-by-wire applications, called Channel A and Channel B, each having a bandwidth of 10 Mbps. These redundant channels also can be used independently and net data rate of the network can be increased to 20 Mbps. Detailed protocol specifications are presented in Section 3.1.

First use of FlexRay technology in standard car production was introduced by BMW in 2007. FlexRay is used for adaptive drive suspension system data exchange in the BMW X5 Sports Activity Vehicle (SAV). Later in 2008 BMW X6 was introduced with fully utilized FlexRay network. It is the first mass produced car using only FlexRay network for in-vehicle communication [16].

CHAPTER 3

FLEXRAY PROTOCOL BASICS: SPECIFICATION, PERFORMANCE METRICS AND SCHEDULING

3.1 FlexRay Protocol Specifications

3.1.1 Media Access Control

In the FlexRay protocol, Medium Access Control (MAC) is based on recurring, identical communication cycles. Each node in the network keeps a cycle counter and the values for this counter are between 0 and 63. Every communication cycle, according to the application specific configuration, can be composed of four different segments. These are *static segment*, *dynamic segment*, *symbol window* and *network idle time* (NIT). Every ECU that is in the same cluster has a common sense of time. This is realized with cluster wide synchronization.

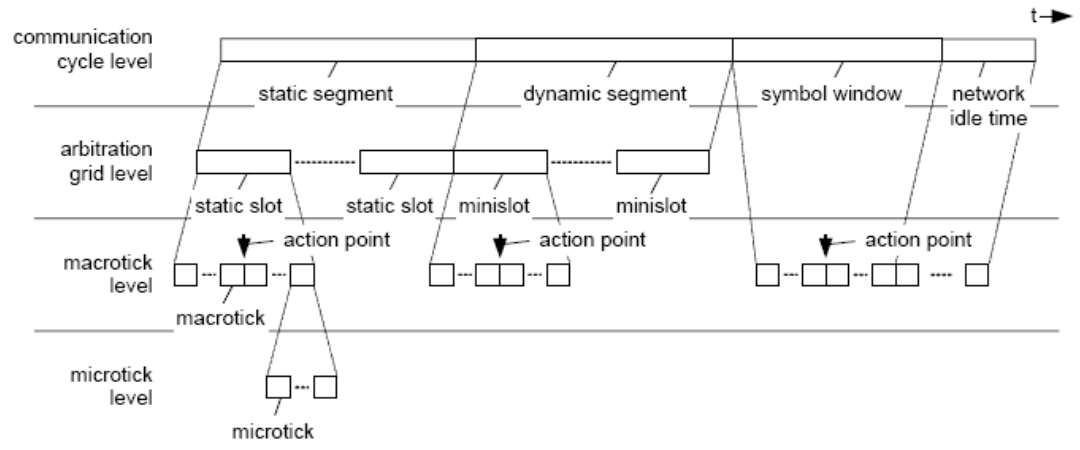

Figure 3-1: FlexRay Communication Cycle Timing Hierarchy [13]

There are four timing hierarchy levels that define the communication cycle. These levels are shown in Figure 3-1.

3.1.1.1 Static Segment

The highest level is communication cycle level and it consists of previously mentioned four segments. The communication cycle always contains a static segment and in static segment a TDMA based Medium Access Control (MAC) scheme is used. Static segment consist of a given number of *static slots.* Number of static slots may change between 2 and 1023, and their durations are all identical for every ECU on the same cluster. The duration and number of static slots are configured prior to the run time. In a specific static slot only one ECU is allowed to send data. However in contrast to TTP/C, FlexRay allows that one ECU can use more than one slot (actually up to 1023) in a communication cycle. This increases the flexibility of the FlexRay. Because of time triggered transmission, delay and jitter characteristics of the messages are deterministic in this segment. Static slot duration is determined by the number of *macroticks* which is a third level parameter in the timing hierarchy. Allowed values for static slot duration are between 4 and 661 MacroTicks (MT). Macrotick is the common time parameter to all nodes in a cluster and composed of ECU specific *microticks,* which is in the fourth level on the timing hierarchy. The number of microticks that constitutes macrotick may change from ECU to ECU. It entirely depends on the local microcontroller oscillator of the ECU. Allocation of static slots to the ECUs is done prior to the execution. So that all ECUs know in which slot(s) it is allowed or not allowed to send data, collisions are avoided by this way. In order to schedule transmissions each ECU has to have a slot counter variable for both channels. Bus guardians are allowed to use in ECUs' communication controllers to increase the fault tolerance of the network. Also "babbling idiot" problem and single point failures do not affect the whole cluster, but only the related ECU. The use of two redundant communication channels is not mandatory. According to the requirements of applications one or two channels can be used by ECUs. However if two are used their cycle parameters have to be same. Some of these parameters are

cycle length, communication cycle level segment durations, static slot and mini slot durations, and macrotick duration. However, these channels can be configured different then each others according to the message scheduling point of view. For example while safety critical applications such as X-by-Wire systems can send the same data in the same slots at each channel, non-safety critical systems can be connected and/or use only one channel. Furthermore one ECU may use third static slot of channel A and fifth static slot of channel B. This flexibility allows the efficient usage of the bandwidth. Figure 3-2 shows the possible static segment message transmission configurations if two channels are used [13].

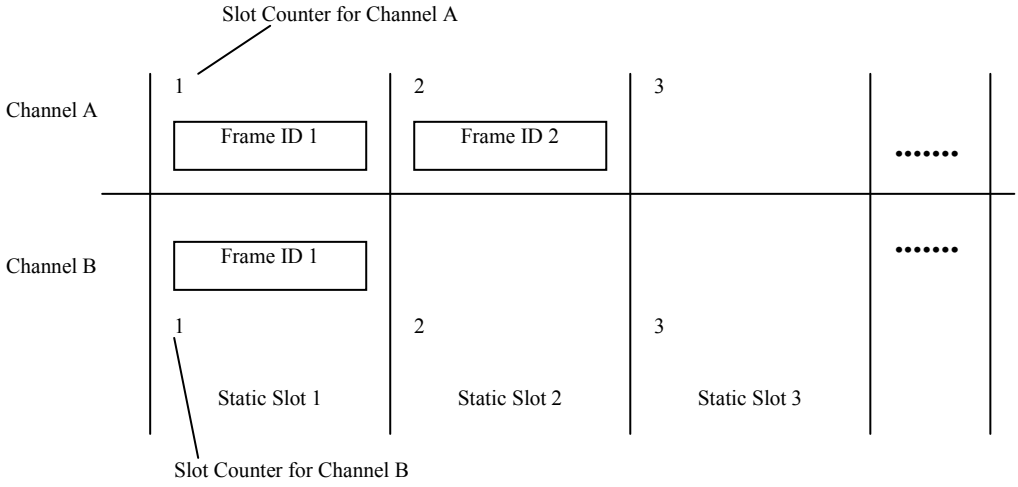

Figure 3-2: Static Segment Message Transmission on Redundant Channels [13]

3.1.1.2 Dynamic Segment

The communication cycle may contain a dynamic segment and this segment is the one where event-triggered message transmission takes place. In the dynamic segment, dynamic minislotting based Flexible TDMA (FTDMA) MAC scheme is used to arbitrate transmissions. It is composed of a configurable number of *minislots*. All minislots consist of identical number of macroticks. If no dynamic segment is required it is possible to configure communication cycle with no dynamic segment.

In dynamic segment message transmission takes place in dynamic slot (communication slot) which consists of variable number of minislots. Furthermore arbitration of messages is based on dynamic slots. Frame Identifications (ID) are used to decide that in which slot the frame shall be sent and arbitration procedure done prior to the run time. Since the frame size that can be sent in this segment is not constant, size and number of the communication slots may also change in order to adapt to the size of the frame. Each ECU has a communication slot counter for both channels. An example transmission scenario is given in Figure 3-3.

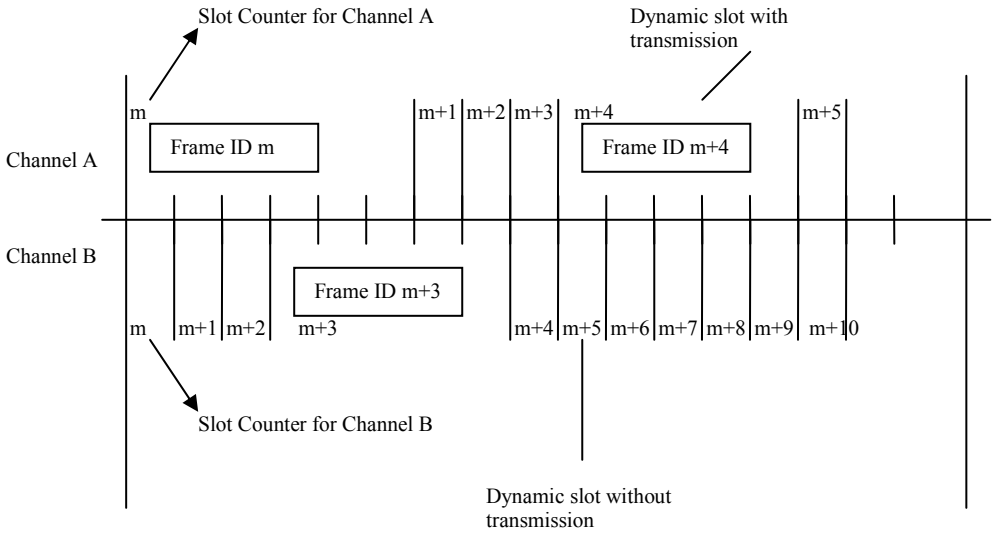

Figure 3-3: Frame Transmission in Dynamic Segment [13]

At the beginning of each cycle communication slot counter value and minislot number are equal. If there is no frame to be sent in the current minislot, slot counter value incremented with minislot number. That means duration of communication slot is equal to minislot duration. On the other hand if there is a frame to be sent then counter value incremented according to another scenario. For example if the duration of the frame is greater than three and less then four minislots duration, then slot counter incremented after completion of frame transmission. This means that while minislot number is incremented by four, slot counter is incremented by one. In this case the duration of the communication slot

(Dynamic Slot) is equal to four minislot duration. This can be formulated as follows:

$$D_{CS} = \left\lceil \frac{D_F}{D_M} \right\rceil \cdot D_M + ProtocolOverhead \quad (2.1)$$

Where;

D_{CS}: Duration of a Communication Slot
D_F: Duration of the Frame
D_M: Duration of a minislot
Protocol Overhead: Defined in the protocol specifications [13].

Different frame transmission arbitration is allowed to be used in channel A and channel B for dynamic segment and this leads to different values for slot counter. Furthermore received frames in the dynamic segment also affect the dynamic slot size. For this reason, while in static segment the slot counters are incremented simultaneously in both channels, they are incremented independently in dynamic segment. The arbitration procedure ensures that receiving nodes know that in which dynamic slot the transmission starts and in which minislot it ends. By this way receiver nodes and transmitter nodes have the same dynamic slot counter value [13].

3.1.1.3 Symbol Window

The communication cycle may contain a symbol window and a single symbol may be sent within the symbol window. The symbol window is used for the network management purposes only. It is a time slot in which the media access test symbol (MTS) can be transmitted over the network [24]. Another application of the symbol window is the bus guardians. It can be used to confirm the normality of the bus guardians if they are used [17]. In a cycle only one ECU can send a symbol and arbitration between different ECUs is not allowed. However it can be realized with a higher level protocol on top of FlexRay. Symbol window consist of a predefined number of macroticks [13].

3.1.1.4 Network Idle Time

The communication cycle always contains Network Idle Time and it consists of predefined number of macroticks. During NIT the ECU calculates and applies clock correction and synchronization with the cluster [13].

3.1.2 FlexRay Frame Format

Structure of the FlexRay frame format is depicted in Figure 3-4. Frames are composed of three main parts; these are *header, payload* and *trailer* segments. Frames are transmitted in the order of as given in Figure 3-4 and individual fields transmitted from left to right.

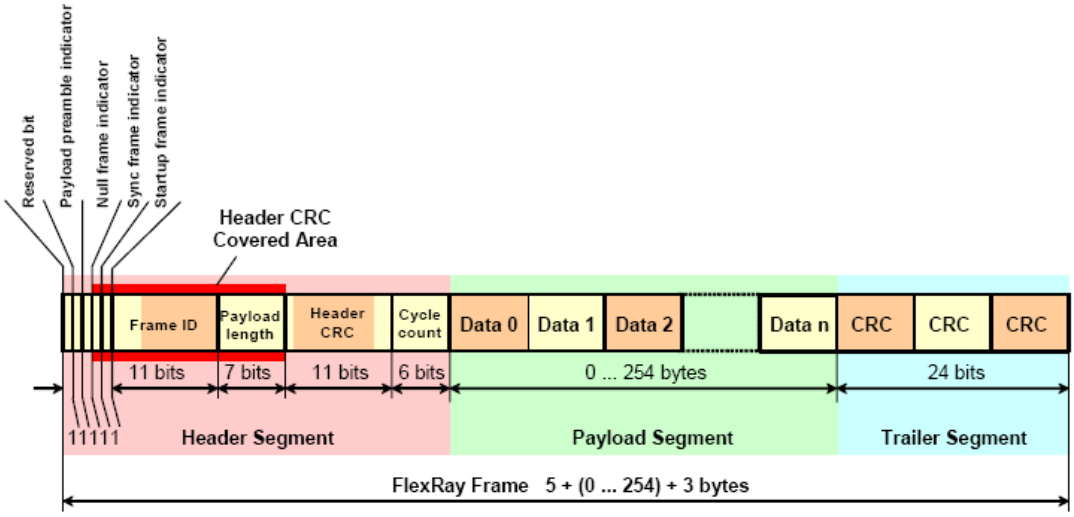

Figure 3-4: FlexRay Frame Format [13]

Frame ID in the header segment defines the slot in which the frame should be transmitted. One frame ID can be used once in each communication cycle. The frames that do not have a Frame ID can not be transmitted in the bus. Frame ID can have values between 1 and 2047. FlexRay payload segment contains data that is multiple of two-byte-words. Maximum allowed payload length is 254 bytes. Furthermore for the messages in the dynamic segment, first two bytes of the payload can be used as Message ID for filtering purpose of the messages. Trailer

segment consist of 3 bytes of Cyclic Redundancy Check (CRC) code for error detection [13].

3.1.3 Composability of FlexRay

The word composability is used to express modular structure or flexibility of the FlexRay protocol. According to the requirements of the domain in which FlexRay is wanted to be used, some properties, given below, that increase the reliability, safety, and fault tolerance can be added or to decrease the cost some properties can be excluded.

High frequency sampling control loops in vehicle dynamic systems, such as chassis control and X-by-Wire systems, demands deterministic and high performance data reception, processing and transmission. Furthermore safety related systems have to satisfy some functional requirements and design criteria. IEC61508 standard defines the requirements for "Functional Safety of electrical /electronic / programmable electronic safety-related systems" and it is used by automotive industry. One of the aims of FlexRay development is to satisfy requirements of future safety and high bandwidth demanded systems. Furthermore to combine several networks in a single network it is developed with a high level of flexibility. FlexRay infrastructure provides several degrees of redundancy and fault tolerance to respond different functional domain requirements [10] [12].

Requirements of safety related systems increase the cost of the network. However, using these high cost networks for non-safety related systems is not wanted. To support the scalable fault tolerance concept FlexRay provides different interconnection topologies such as, single channel, dual channel or mixed channel. Furthermore while non-critical ECUs can be used without bus guardians, in a safety-critical system all ECUs have a bus guardian to increase the reliability. FlexRay also allows simple bus, star and multiple star to be used for limiting the fault propagation. Apart from the topological flexibility clock synchronization algorithms also can be designed as fault-tolerant or non fault-tolerant [10] [12].

If automotive manufacturer suppliers' role is considered, FlexRay supports independent deployment strategies of ECUs. Structural and functional domains are completely separated from each other. It is not important for a subsystem that how the entire system topology is constructed. It only deals with its own functionality. A single ECU does not need to know about the structure or schedule of the whole system. It only knows in which cycle and slot it is allowed to send and it will receive data. If one ECU is added to the system only related ECUs, which will transmit data to the new ECU or receive data, needs to be updated. If the new ECU requires higher bandwidth than a single static slot can supply, more than one static slot can be assigned. This feature of the FlexRay makes it easy to update networks when new components are added. In event triggered networks whole system must be redesigned and extensive testing must be performed for new components [10] [12].

3.2 FlexRay Network Performance Metrics - Static Segment

Performance evaluation of FlexRay network is carried out according to defined performance metrics. These metrics are applied to designed experiments and the results are evaluated. Two different segments, static and dynamic, of the protocol have different characteristics. Therefore metrics are separated as static segment and dynamic segment metrics.

3.2.1 Static Slot Allocation (SA)

Allocation means that the specific static slot is reserved for a specific message for a specific cycle. Any other message, apart from the one that the slot is reserved for, can not use the slot in that cycle. *SA* is the total of actually allocated portion of the static slots for the message transmission in one cycle.

Periods of the messages can be much longer than FlexRay cycle time. Assume that cycle duration of the network is 5 ms, and period of the message *m* is 500 ms. In this case message *m* will be generated once in every 100 cycles. This number shows the number of cycles in which the message is repeated once, and is called as the *Repetition Time of the message m.* It is defined as

$$m_{RTm} = \frac{P_m}{gdCycle} \tag{2.15}$$

Where *Pm* is the period of message *m* in terms of second, *gdCycle* is the cycle duration in terms of second, and m_{RT} is the repetition time of the message *m*. Assume that a message *m* is assigned to be sent in a static slot and *repetition time of the frame* which carries message *m* is f_{RTm}. This frame is the one that the message can be transmitted inside. If the message is ready at time of allocated slot than the frame is sent with the message inside and if it is not ready the frame is sent empty. If $f_{RTm} = 1$ (frame is transmitted in every cycle) then 99 times in every 100 cycles the frame is sent empty and related bandwidth will be lost. To avoid this bandwidth loss FlexRay protocol allows that the static slot can be allocated only in the necessary cycles and the frame is sent only in allocated cycles by setting $f_{RTm} = m_{RT}$. For the remaining cycles slot is free and can be allocated to other messages of the same node. For example in the above case setting f_{RTm}=100 prevents bandwidth loss. Message waits in the buffers until the time for frame transmission. For the given reason f_{RTm} should have the same value with m_{RTm}; however in some situations (these situations are expressed in Section 5.1.2) they can take different values.

Frame repetition time shows the number of cycles between transmissions of two consecutive frames or another way to say it is the frame period (P_{fm}) in terms of FlexRay cycle duration. Assume that the period of the frame is P_{fm} and FlexRay cycle duration is *gdCycle*, then f_{RTm} is calculated as

$$f_{RTm} = P_{fm} / gdCycle \tag{2.2}$$

Consequently $1/f_{RTm}$ is defined as the Static Slot Allocation for the message *m* (SA_m) and formulated as

$$SA_m = 1 / f_{RTm} \tag{2.3}$$

Finally, if M is total number of messages in the static segment SA for the Static Segment is given as

$$SA = \sum_{M} SA_m = \sum_{M} 1/f_{RTm} \tag{2.4}$$

The unit of the SA is (*static slots/cycle)*. For the configuration of a specific message set having smaller SA values is better as compared to larger values of SA. Small SA value for a given message set means that same message set are transmitted by allocating less number of static slots.

3.2.2 Used Static Slot ID (U_{ID})

Apart from the SA, used static slot IDs is another important measurement for the allocation. If a slot is allocated for a given message whose repetition time is greater than one the slot is not fully allocated to this message. However, that slot ID can be used only for that node, which transmits the message. That means, the slot is partially utilized and can not be used by any other nodes. For this reason using less slot IDs allows for a large number of free slot IDs for the future new nodes without changing the static slot configuration. If a given Slot ID is allocated for message transmission either fully or partially it is counted as used. Then

$$U_{ID} = Used\ number\ of\ Static\ Slot\ IDs \tag{2.5}$$

Note that $U_{ID} \geq SA$.

There is a limit for the possible values of U_{ID}. At least one static slot will be used for each node in a given cluster, and then minimum number for U_{ID} is equal to number of nodes (N). Furthermore used U_{ID} can not be larger than total number of available static slots (T_{SS}). Then the boundaries for possible values given as

$$N \leq U_{ID} \leq T_{SS} \tag{2.6}$$

3.2.3 Utilization (U)

Utilization is defined as the ratio of used bandwidth to allocated bandwidth and given as

$$U = \frac{B_U}{B_A} \tag{2.7}$$

Where U stands for utilization, B_U is the used number static slots in one second, and B_A is the allocated number of static slots in one second. Larger utilization shows that configuration of the network is constructed to use the bandwidth more efficiently.

In our experiments 10 Mbps baud rate is used as the FlexRay data rate. The calculation of the utilization in our experiments is performed as follows:

SA is the allocated static slots in one cycle and each cycle in a given static slot only one message can be transmitted. FlexRay cycle is repeated

$$1/CycleLength \tag{2.8}$$

times in one second. Where, the unit of the *CycleLength* is in seconds. Then

$$B_A = SA*(1/CycleLength) \tag{2.9}$$

gives us the number of messages that are scheduled to be sent (capacity allocated for) in one second. Another way of expressing the allocation is the ratio of allocated time duration in a single cycle for message transmission in the static segment to duration of a cycle. This ratio is called as Allocated Bandwidth Ratio (BW_A) and given as,

$$BW_A = (SA*gdStaticSlot*gdMacrotick)/gdCycle \tag{2.10}$$

where, the meanings and the units of the variables *gdStaticSlot, gdMacrotick* and *gdCycle* are given in Section 5.1.2.

Used number of static slots is calculated from the received data. The number of received messages in one second gives the used number of static slots because every message is transmitted exactly in one static slot.

$$B_U = NumberOfReceivedFrames\ (nonempty)\ /\ seconds \qquad (2.11)$$

Bandwidth usage can be expressed in another way to show the actual used capacity of the FlexRay in terms of bits per second (bps). This expression is calculated as follows

$$BW_U = 10\ Mbps * BW_A * U \qquad (2.12)$$

where, BW_U is the used bandwidth in terms of bits per second (bps), BW_A is the allocated bandwidth, U is the utilization and 10 Mbps is data rate of the FlexRay network.

3.2.4 Jitter (J)

In the control applications, the control algorithm works better and therefore performance of the controller is increased if the input data is available periodically. For this reason apart from the generation of messages periodically, the receiver nodes need to get these messages also periodically. Jitter is defined as the time variation in the reception of periodic messages. Therefore having jitter values as low as possible increases the controller performance of the electronic applications. Another definition of the jitter is the deviation of the time difference of two consecutive instances of a message from the message period and given as

$$J_{M(k)} = \left| P_M - (T_{TM(n)} - T_{TM(n-1)}) \right| \qquad (2.13)$$

Where J_{Mk} is an instance of the jitter for the message M, P_M is the period of message M, and $T_{TM(n)} - T_{TM(n-1)}$ is the time difference between two consecutive instances (n^{th} and $(n\text{-}1)^{th}$) of the message M. Note that K (total number of jitter values) = N (total number message instances) - 1. Because of the bandwidth

allocation in the static segment, message transmission with zero jitter can be achieved with FlexRay protocol.

3.2.5 Delay (D)

For safety critical systems generated messages have to be transmitted immediately to the receiver nodes. The time difference between the transmission and generation of an instance of the message *M* is defined as delay and given as

$$D_{M(n)} = T_{TM(n)} - T_{GM(n)} \tag{2.14}$$

Where $D_{M(n)}$ is the delay for the n^{th} instance of the message *M*, $T_{TM(n)}$ is the time when the instance of the message is transmitted and $T_{GM(n)}$ is the time when the instance of the message is generated [21]. Keeping the delay smaller than the period of the message is critical for the electronic applications. For periodic messages, period value is considered to be its deadline. In our experiments both the maximum and average delay values are calculated. Maximum delay shows the worst message transmission which has to be less than the deadline for the system to work properly. The average delay shows the general tendency of delay values for the given message ID.

3.3 FlexRay Network Performance Metrics – Dynamic Segment

3.3.1 Delay (D)

Dynamic segment is the part of the communication where sporadic messages are transmitted. Timings of these messages can not be known prior to run time. Therefore we would like to know if the messages are transmitted on time. Dynamic messages have a deadline value, which is defined as the maximum permissible transmission time. Messages must be sent before deadline has passed. A missing deadline of the message may lead to undesired consequences. The message delay for the dynamic segment is computed in the same way as for the static segment as in (3.13).

3.3.2 Deadline Miss Ratio (DMR)

Each real-time message has a certain deadline value that limits the maximum delay for the message. To exceed this value can cause unexpected results according to the properties and requirements of the application. The ratio of messages that are missed their deadline to the total number of messages transmitted is defined as the Deadline Miss Ratio (*DMR*) and given as

$$DMR = \frac{NumberofMessagesWithMissedDeadlines}{TotalNumberofTransmittedMessages(nonempty)} \tag{2.15}$$

Note that this metric is investigated for the sporadic messages in the dynamic segment only. The deterministic slot allocation in the static segment guarantees that the messages meet their deadlines. Constructing a schedule for the dynamic segment which guarantees that the deadlines are met is investigated in [26]. This approach requires software architecture to be implemented in the ECUs hence its implementation is not within the scope of this thesis

3.4 Message Scheduling for the FlexRay Network – Static Segment

Deciding the transmission time and the transmission method of messages prior to the run time (offline) is called as scheduling. The schedule for a message is an important parameter that affects the performance metrics as described in Section 3.2. While performing the scheduling of the static segment messages, the approaches in [21] are used. These scheduling approaches can be divided into two categories. In the first one, messages are transmitted without jitter and in the second one; messages are transmitted allowing a certain amount of jitter. Furthermore message packing for messages with the same periods can be applied to increase the utilization of the network by decreasing the overall framing overhead. Message packing is defined as combining two or more messages into one FlexRay frame.

As mentioned in Section 3.2.4 jitter is an important parameter and keeping the jitter as low as possible increases the performance of the control applications.

Furthermore increasing the utilization of the network and decreasing the U_{ID} and *SA* is other important concerns as given in performance metrics. Therefore the methods that are applied while constructing the scheduling should improve the performance of the FlexRay network.

Before explaining the methods in [21], *cycle filtering* and *cycle multiplexing* concepts should be defined. The scheduling in the static segment is done according to f_{RT} values of the frames that the messages can be sent inside as explained in Section 3.2.1. There is one more additional parameter that decides the allocated slots for the messages called *offset. Offset* value shows the starting cycle counter value for the message allocation and it is less than f_{RTm}. Assume that for a message *m*, $f_{RTm} = m_{RTm} = 2$ and *offset* = 1. Then the allocation of the static slot to the message *m* over 64 FlexRay cycles is set as shown in Figure 3-5. In the figure colored boxes shows allocated cycles. If the *offset* value is 0 then allocation would start from the cycle 0. In the run time Communication Controller checks the current value of the cycle counter and the values of the f_{RT} and *offset,* if there is a match the frame is transmitted with the message. This feature is called as the *cycle filtering*.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 3-5: Allocation of the static slot to the message m over the cycles 0 to 63
f_{RT}=2, *offset* = 1

When cycle filtering is applied the static slot will be empty in some cycles. Empty cycles can be used by the same node for other messages again using the cycle filtering. In this way same slot can be allocated to different messages of a node in

different cycles. This feature is called as the *cycle multiplexing*. In [21] the methods explain how to do a scheduling by using cycle filtering and cycle multiplexing and a formulation is given for the optimum number of U_{ID} for both of the case, scheduling with jitter and without jitter. Number of frames that belongs to a specific node, their repetition times and the messages to be transmitted with the same slot ID are the parameters used for this optimization.

Figure 3-6 shows an example in which one static slot is allocated for 4 different messages that are generated by the same node, each with $m_{RT} = 4$ assuming $f_{RT} = m_{RT}$.

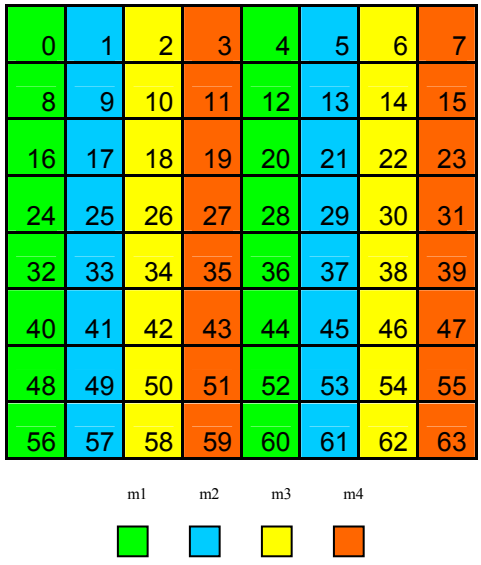

Figure 3-6: Full Allocation of a static slot for 4 messages by using Cycle Multiplexing

In this figure message m_1 has a repetition time of 4. It is generated and transmitted in every 4 cycles. In the other 3 cycles, the slot can be allocated to messages m_2 and m_3 and m_4. The message with lower offset will be sent in low numbered cycles. In this example the offset values are 0, 1, 2 and 3 respectively. If cycle filtering and cycle multiplexing are not used then we would need 4 slots for 4 messages and each of these 4 slots would be fully only utilized by 25%. By this method utilization of the bus is increased to 100% and U_{ID} is reduced to 1 and also jitter is

zero for all the messages. Although cycle multiplexing is applied, there may be still empty cycles depending on the message set properties. Figure 3-7 shows such a situation.

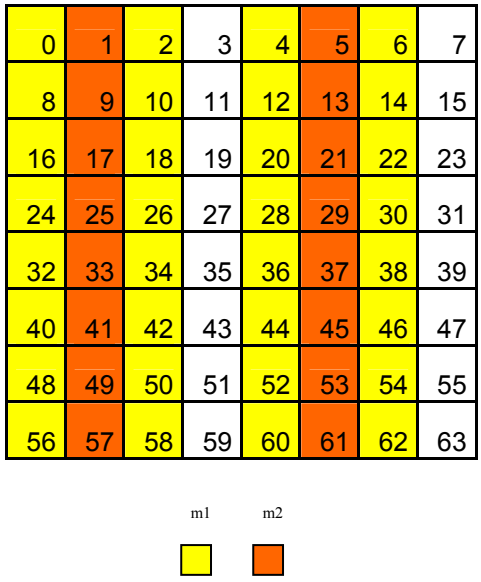

Figure 3-7: Partial Allocation of a static slot for 2 messages by using Cycle Multiplexing

In the figure m_1 has a repetition value of 2 and *offset* of 0, m_2 has a repetition value of 4 and *offset* of 1. As can be seen from the figure slot is empty once in every 4 cycles. This slot can be further allocated to a message with repetition time of an integer multiple of 4 (without jitter case) or with a repetition time of not an integer multiple of (with jitter case) [21]. If the repetition time of the new message is less than 4, some slots will be allocated to more than one message in the same cycle. This violates the protocol rules and has to be avoided.

In Figure 3-6 and Figure 3-7 repetition times of the messages have a common divisor greater than 1. This situation satisfied by choosing the cycle duration value as the greatest common divisor (gcd) of the message periods. Therefore the jitter is zero. However in some situations zero jitter may not be possible with cycle multiplexing. In these situations either an extra slot need to be used or the jitter has to be allowed [21].

CHAPTER 4

DEVELOPMENT TOOLS USED FOR THE EVALUATION OF THE PROTOCOL

The evaluation of the FlexRay is performed in the real time with a hardware environment. These tools are designed and manufactured very recently and specific to the FlexRay protocol. One of the early steps of this thesis work is exploring these new tools and learning their working structure. Better understanding of the constructed network and performed experiments can be realized by looking at the functional structure of the tools. For this reason, the hardware and software tools that are used in the implementations of experiments are presented in this section.

4.1 SK-91465X-100MPC Fujitsu FlexRay Evaluation Board

The SK-91465X-100MPC is a multifunctional evaluation board for the Fujitsu 32-bit Flash microcontroller series MB91F465XA (CPU). It is the main building block of the experiments and used as the ECU that generates and transmits messages with certain timing properties. It supports FlexRay protocol operations and has two redundant channels as Channel A and Channel B. Protocol operations are implemented with two Bosch E-Ray IP-Module types Communication Controllers (CC). CPU of the evaluation board is used for configuration of this CC and manipulation of the messages. Generated messages are sent to CC transmit buffers and received messages from the bus are stored in the CC receive buffers. These received messages can be read by the CPU. Transmission of the buffered messages is performed by CC according to preset configuration parameters.

The connection to the physical layer of the FlexRay bus is realized with AMS8221B transceiver. Apart from the FlexRay channels, Evaluation Board also has 2 CAN interfaces for the evaluation of CAN protocol, 2 LIN/UART interface for the Evaluation of LIN protocol and an UART interface for the communication with CPU. It can be used for the Gateway development because of the multiprotocol support [17]. A representation of the Evaluation Board is given Figure 4-1.

In addition to the hardware, evaluation board package also includes a software development environment, a flash programming tool, FlexRay Communication Controller Driver, C-Compiler, Assembler and Linker. General description and properties which are used during the experiments of these tools are given in the following sections [18].

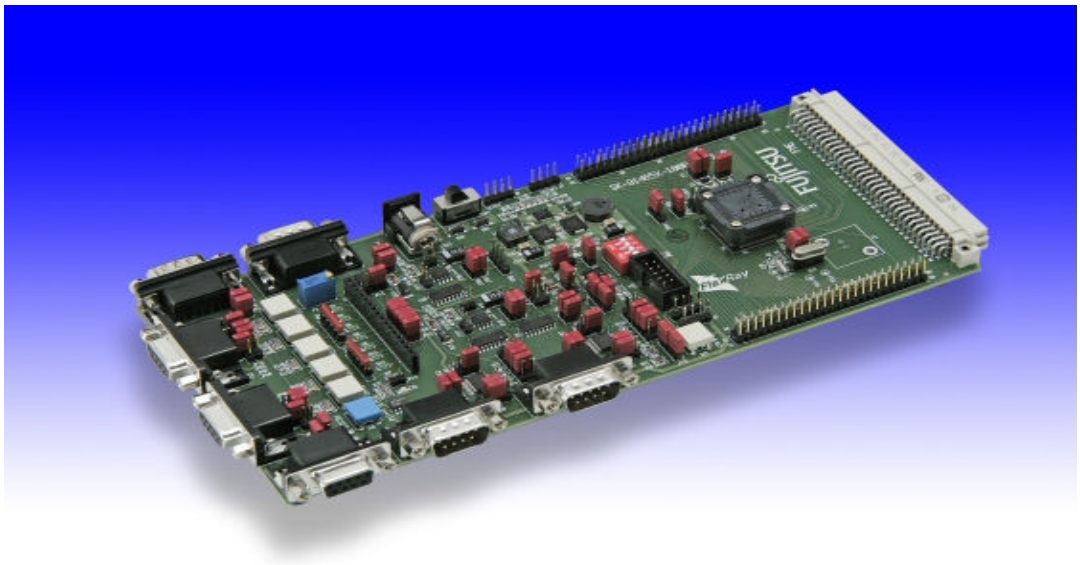

Figure 4-1: SK-91465X-100MPC Fujitsu FlexRay Evaluation Board

4.1.1 Softune Workbench Software Development Environment

Softune Workbench is a 32 bit software development environment supplied in the evaluation board package. It allows the creation, storage and easy manipulation of

software projects. Developed software codes are compiled and "*.mhx" type files which are loaded to microcontroller is created. In addition to the "*.mhx" files, the "*.abs" files which includes the debugger necessary information and machine code are also created by compilation. The user interface of the Softune Workbench is given in Figure 4-2 [18].

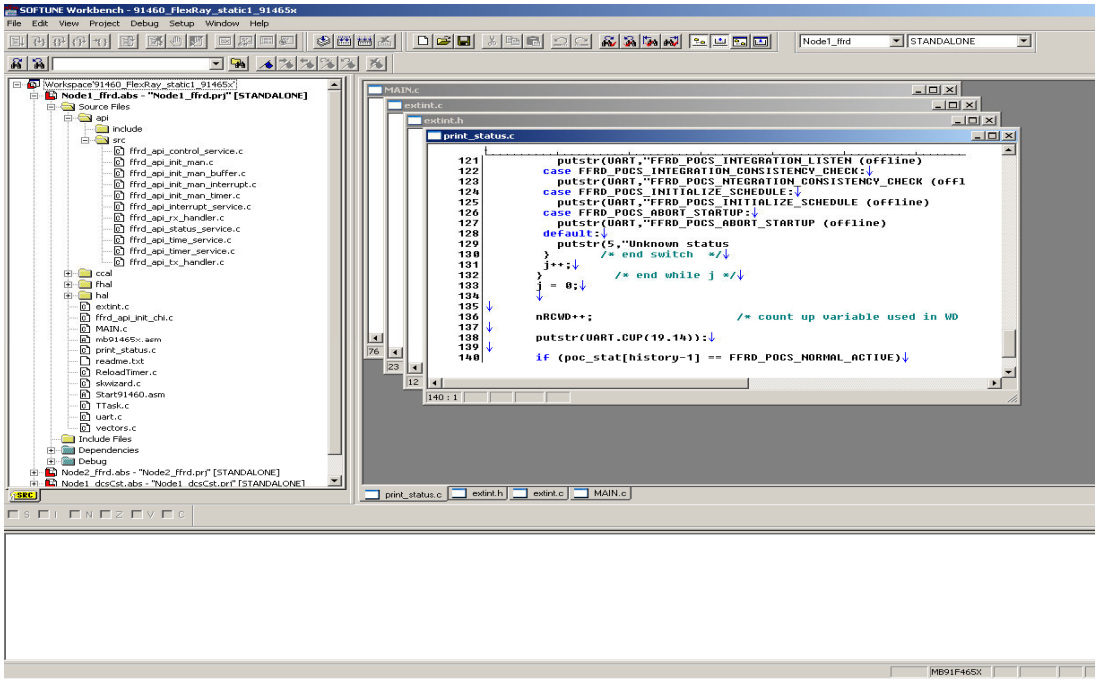

Figure 4-2: Softune Workbench user interface

4.1.2 Flash Programming Tool

It is used for the loading of "*.mhx" files to microcontroller flash memory. USB, COM or Ethernet ports of the Personal Computer (PC) can be used for the communication with Microcontroller. In our experiments FME FR Flashprogrammer V4.2 is used. Figure 4-3 shows the user interface of the tool [18].

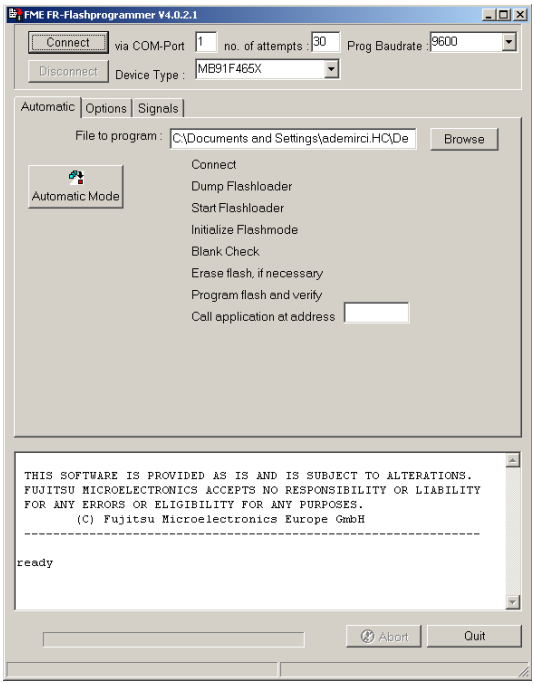

Figure 4-3: Flash Programmer user interface

4.1.3 FlexRay Communication Controller Driver

Within the evaluation board package there is a FlexRay driver which enables easy implementation of FlexRay applications without the need for dealing with the specific register addresses. It includes also source code for development purposes. Configuration of the communication controller is also done by the driver application programming interface. It can be performed manually or choosing the Code generators supporting "*.chi" based output files which explained in the Section 4.2.

Driver code consists of 4 layers. These are the driver layer which contains Application Programming Interface (API), Communication Controller layer (CCAL) for the functionality of the FlexRay driver, FlexRay Hardware Layer (FHAL) which includes all functions for the used FlexRay hardware and Hardware Layer (HAL) which includes all functions for the dedicated used hardware. API layer is the user interface of the FlexRay driver. CCAL layer includes the routines for the driver. FHAL defines the FlexRay hardware descriptions such as address

offsets. HAL layer consist of microcontroller read and write operations. Implementation of the driver is realized in the C-Language and supports only Bosch E-Ray communication controller. This layered structure is presented in Figure 4-4.

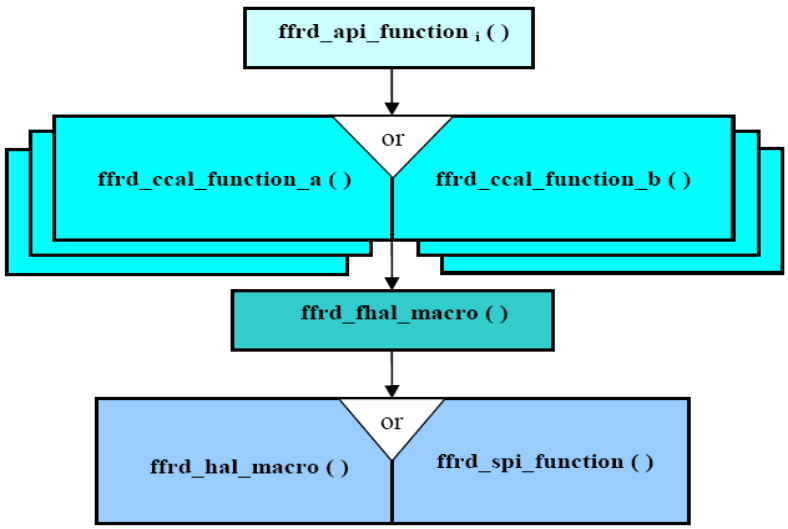

Figure 4-4: Architecture of the FlexRay Driver

When an application calls one of the API functions this function evaluates the call parameters and calls related CCAL routine. These routines include computing values, register settings, buffer requests and interrupt routines. CCAL routine calls the macro from FHAL layer and E-Ray address offset is added. Finally at the HAL layer Microcontroller and Communication controller access is realized.

A specific application may not need to use all of the functions that are available in the API layer. The services of the API layer are divided into functional domains. Each service domain fulfils only the related functions. The reason for this division is to avoid unnecessary inclusion of the functions. Only necessary services are included in the code of the applications. In the deriver code including of a service is performed by making the necessary setup in the “ffrd_api_global_def.h” file. This file also includes other global settings of the driver and has to be examined before starting an application development. By this way the size of the

microcontroller code can be kept relatively small. The list of the functional services is:

- Initialization Services (by "*.chi" files or manually)
- Control Service
- Interrupt Services
- Reception (Rx) Services
- Status Information Services
- Time Services
- Timer Services
- Transmission (Tx) Services

The principle program flow of the driver and usage of the API services are given in Figure 4-5. After Reset Init services and part of the Control and Status services are available and other services are available after the initialization is completed [19].

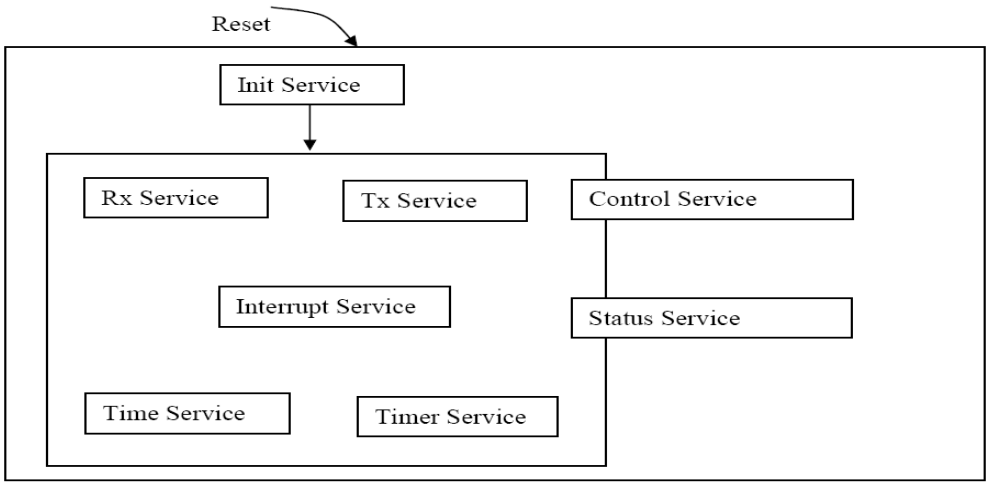

Figure 4-5: Program flow of the FlexRay Driver

4.1.4 C-Compiler, Assembler and Linker

C-Compiler, Assembler and Linker are used with Softune Workbench environment. Generated text code is compiled and linked with these tools and made available for downloading to microcontroller.

4.2 FlexConfig™ Developer – Universal FlexRay Configuration Tool

Flexible structure of the FlexRay protocol provides many configurable parameters. Since the number of these configuration parameters is large, settings of the values have to be carefully examined according to protocol specifications. A universal configuration tool, FlexConfig™ Developer version S3V0-F, is used to decrease the configuration errors to zero during the experiments. The configuration parameters of the protocol are entered through the user interface of the FlexConfig tool. Then the tool checks all the parameters and displays error messages and suggestions for nonconformances if there is any. Created configuration file can be exported as "*.xml" file. Furthermore configuration parameters of each node can be exported as controller host interface ("*.chi") files. Later these "*.chi" files can be included in the microcontroller code and initialization of the communication controller of the evaluation board is done by included information without any error. The user interface of the FlexConfig tool is given in Figure 4-6.

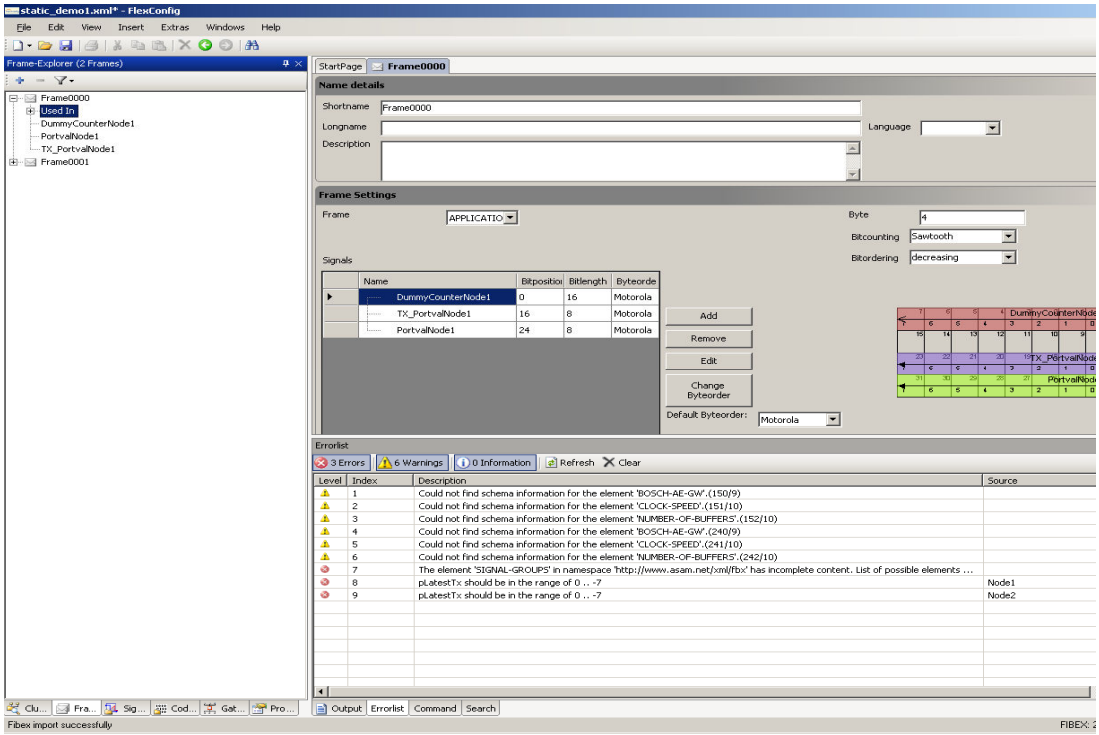

Figure 4-6: FlexConfig user interface

4.3 FlexCard Cyclone II SE

FlexCard Cyclone II SE is a measuring instrument for analysis of the FlexRay protocol and it can be also used to send data to other FlexRay bus members. In our experiments FlexCard is defined as the receiver of the all messages sent by other bus members (Evaluation Boards). It is a 32 bit CardBus card which is supporting 2 redundant FlexRay channels as Channel A and Channel B and also it has 2 high speed CAN channels. Quick access times and high data throughput of the tool is the additional features that is important while evaluating the high data rate of the FlexRay protocol. Furthermore exact time stamp of the received messages can be achieved by the card and this feature allows accurate timing calculations such as delay and jitter of the messages. A picture of the tool is given in Figure 4-7 [20].

Figure 4-7: FlexCard Cyclone II SE

4.4 FlexAlyzer

The FlexAlyzer is a software tool that is designed for FlexCard Cyclone II SE to monitor and analyze the bus traffic of the FlexRay network in which FlexCard is

connected. The time information of the received messages can be viewed through the user interface of the tool. Monitoring can be performed in two modes which are synchronous and asynchronous. In the synchronous mode, user has to include related controller host interface file. In our experiments FlexCard is taken as a separate ECU and defined as the receiver for all messages in FlexConfig. Then "*.chi" files are created and included in FlexAlyzer. Synchronous mode allows more accurate monitoring and analyzing of the bus traffic with respect to asynchronous mode. Time information includes time stamp, received cycle, slot ID, FlexRay channel, payload data and status information of the received messages. Log data of the received messages can be stored as text files to be later analyzed. The user interface of the tool is given in Figure 4-8 [20].

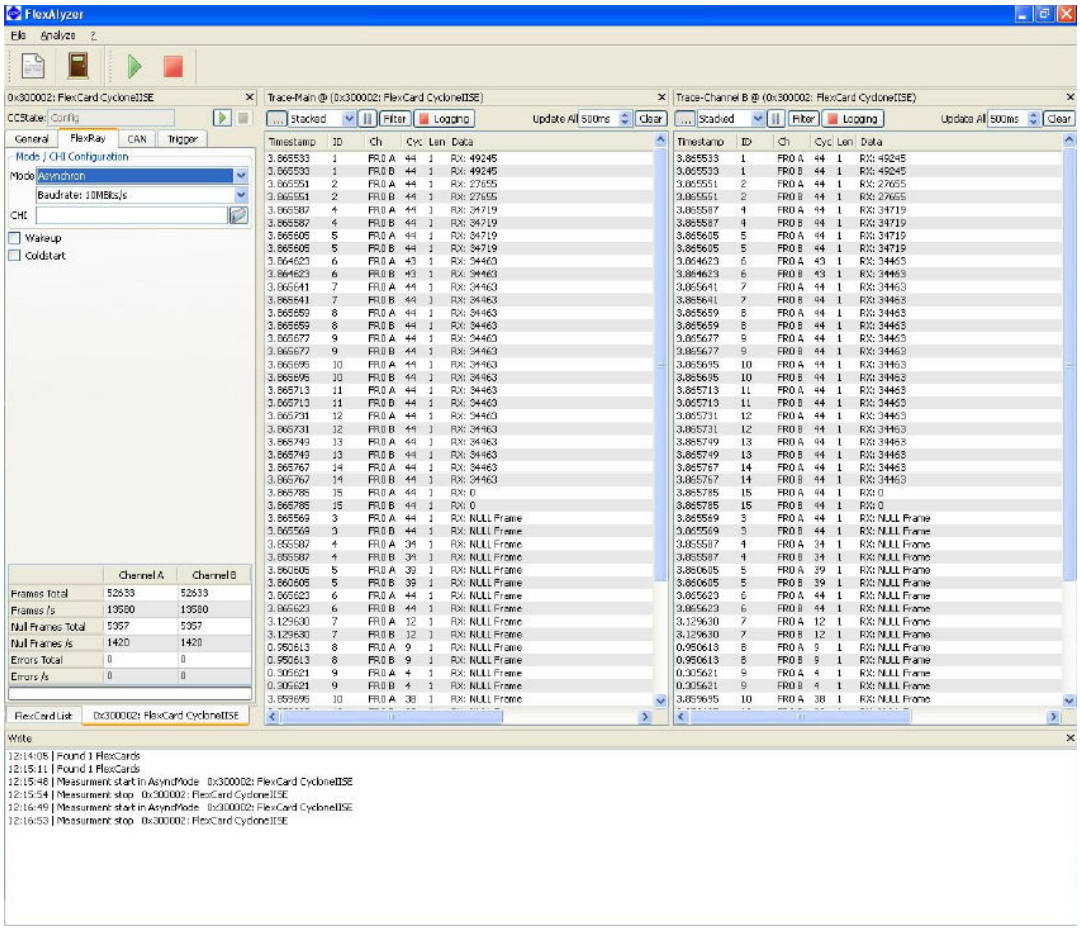

Figure 4-8: FlexAlyzer User interface

CHAPTER 5

DESIGN OF EXPERIMENTS

5.1 Work Flow in the Experiments

During the design and implementation of experiments the work flow that is shown in Figure 5-1 is followed. The details about each step are given in the related section.

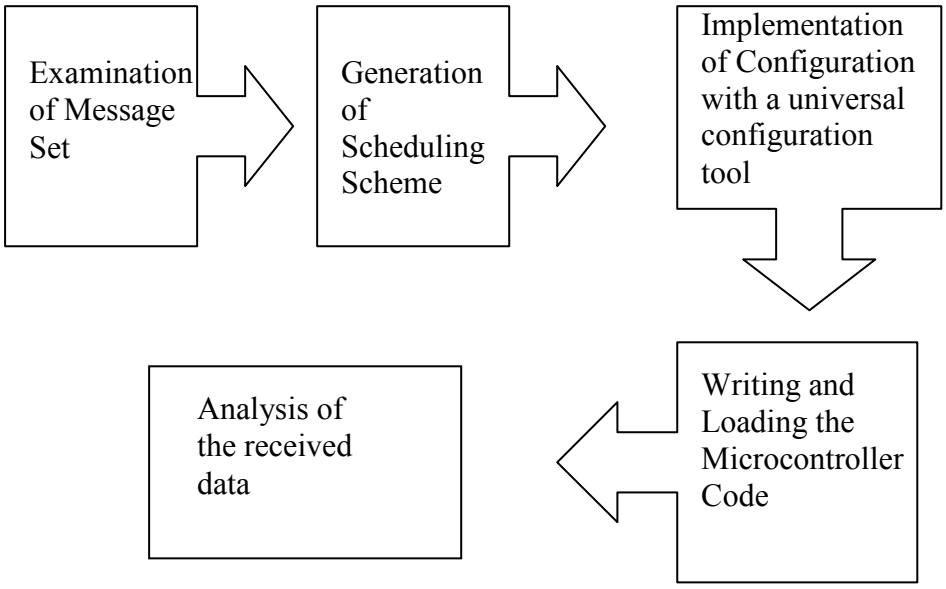

Figure 5-1: Work Flow for the Experiments

5.1.1 Examination of Message Set

The parameters that are included in the message set are Message ID, Message Type (periodic/sporadic), Message Period, Message Size, and Sender and Receiver Nodes. These parameters are used for designing the configuration of the network

and constructing the scheduling table. The structure of the message set is given in Table 5-1.

Table 5-1: Message Set Table

Message ID	Message Type	Message Period	Message Size	Sender Node	Receiver Node
Message Identification	Periodic(P)/ Sporadic(S)	in ms	in bytes	Node Name	Node Name

Message ID is used to identify the message. In static and dynamic segments this variable can be used to assign priorities.

Timing behavior of the message is defined by its *type.* There are two types of messages, periodic and sporadic messages. Efficient use of bandwidth requires that periodic messages are sent in static segment, and sporadic messages are sent in dynamic segment.

For periodic messages *Message Period* is defined as the time difference between two consecutive messages and its value is constant through time. On the other hand it means minimum time difference between two consecutive messages for sporadic ones. This time difference during the experiment can be much longer than the period value because of its random nature.

Message Size is the actual payload data sent through the network. Sender Node is the node that sends the message and Receiver Node is the intended receiver of the message.

5.1.2 Generation of Scheduling Scheme

To indentify the needs for the cluster wide configuration parameters, message specific configuration parameters need to be defined first. This step includes the determination of *repetition times of the frames* and allocation of static and dynamic slots. Static and dynamic segments are independent from each other. Therefore

allocation of periodic and sporadic messages also can be seen as independent from each other.

Scheduling scheme (message allocations) is based on the categorization of messages according to message set parameters. Top two categories in this scheme are static segment and dynamic segment messages. This choice is made according to the message types. While periodic messages are sent in static segment, sporadic messages are sent in dynamic segment. For static segment further scheduling of the messages are performed. The scheduling places the periodic messages into static slots by considering the previously defined performance metrics. Further placement of sporadic messages into dynamic slots is done according to their Message IDs.

If all the nodes that are using static segment generate periodic messages synchronously with FlexRay cycles (message generation at the beginning of FlexRay cycles), then static segment scheduling also can be done according to message priorities. The messages that are assigned to be high priority can be allocated to low numbered static slots. Synchronized message generation scheme is presented in [23]. Since all the messages generated at the same time the one that has the lowest Static Slot ID (highest priority) is transmitted first and delay is low. However this can be realized only if message periods are integer multiples of the cycle time and this situation is satisfied by choosing such a FlexRay cycle duration. This priority assignment does not affect the jitter of the messages but only delays because consecutive messages will be transmitted in the same Static Slot ID. Figure 5-2 shows a situation that three static segment messages are created at the beginning of FlexRay cycles and repetition times of the frames for that messages are set to 1. Then while the message with ID 1 (highest priority) will experience the shortest delay, ID 3 (lowest priority) will experience the largest delay.

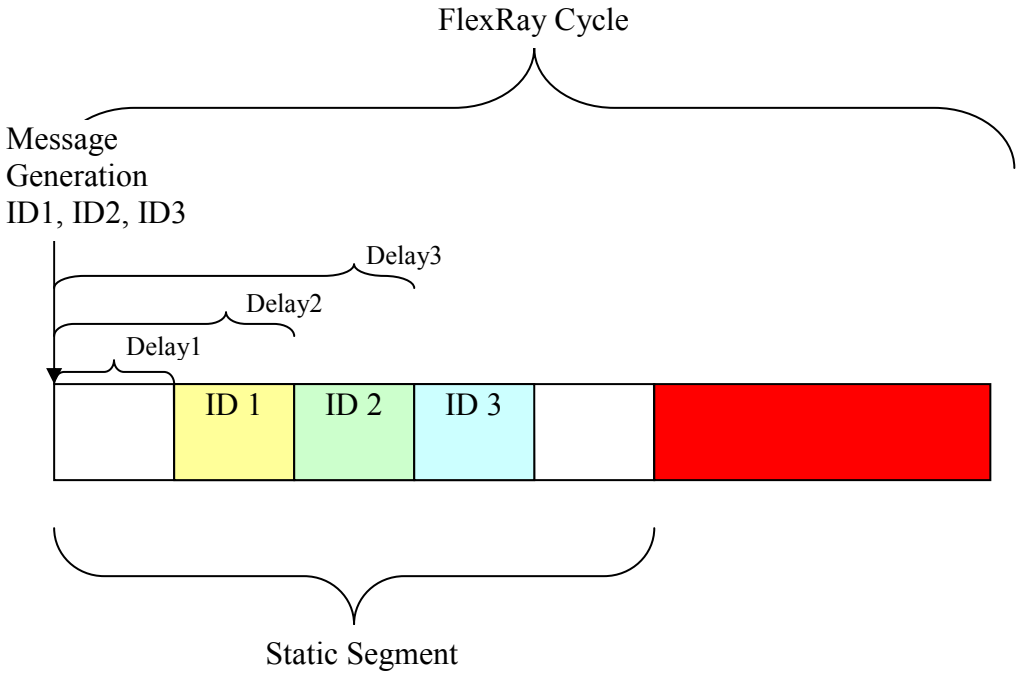

Figure 5-2: Static Segment Priorities

In the experiments, construction of the message scheduling with the simplest scheduling method (one message per static slot) that is given in the Section 6.1.1 is applied first. Then the methods given in Section 3.4 are applied to improve the performance metrics. However those methods assume that repetition times of the frames can take any integer values. Therefore some modifications are applied to adjust the methods according to capabilities of our tools.

The implementation of the Communication Controller (CC) of the Evaluation Board only allows the values of 1, 2, 4, 8, 16, 32, and 64 for the f_{RT}. For this reason repetition times of the frames (f_{RT}) that the messages transmitted inside are taken different than actual repetition times of the messages (m_{RT}). If the implementation allows all integer values then these two repetition time values would be equal. However f_{RT} is changed because of the implementation of CC and the reasons are given below.

It has to be considered that f_{RT} has to be less than or equal to the m_{RT} if deadline of the message is equal to period. Otherwise f_{RT} has to be less than *deadline/gdCycle*.

The modified method for the minimum jitter is as follows: Assume that a message *m* has a period of 2000 ms and the cycle duration (*gdCycle*) is 5 ms. Then m_{RT} is calculated as 2000/5 = 400. Therefore static slot need to be allocated to the message *m* once in every 400 cycles. Due to the limits of the CC, f_{RT} is set to the largest power of two divider of the m_{RT} to satisfy the minimum jitter for the messages. For the message *m* dividers of 400 is

$$400 = 2*2*2*2*5*5 \tag{4.1}$$

Then largest divider which is a power of two is 16. Therefore f_{RT} is set 16 in the CC configuration. By this way 400^{th} cycle, that the message need to be transmitted, will be allocated and jitter is minimized. Moreover the utilization is increased by freeing slots in the unused cycles. However 24 cycles out of 400 will be empty again and wasted.

This method is shown with an example. Assume that the message *m* is wanted to be assigned to a static slot and $m_{RT} = X$, then static slot allocation for this message $SA_m = 1/X$. Keeping this in mind assume a message set with four messages. Periods of the messages are P_{m1}=2000ms, P_{m2}=1000ms, P_{m3}=500ms, and P_{m4}=100ms and the cycle duration for the cluster is 5 ms. m_{RT} values are 400, 200, 100, and 20 respectively and then f_{RT} values are 16, 8, 4, and 4 respectively for the minimum jitter case. Then the sum of *SA* values of each message is:

$$SA = 1/16 + 1/8 + 1/4 + 1/4 \approx 0.69 < 1 \tag{4.2}$$

The sum is less then 1, which means one static slot is enough to schedule all four messages. Furthermore periods of the messages are not coprime. This means that all messages can be scheduled without jitter. With the suitable offset values that prevent collision of the messages (collision means that allocation of the slot to more than one messages at the same cycle); message scheduling can be constructed

as shown in Figure 5-3. This figure shows the allocation of the static slot over 64 cycles. Colored boxes show the allocated cycles and white boxes show the free cycles. The numbers that are inside the boxes indicates the cycle counter value of the FlexRay nodes.

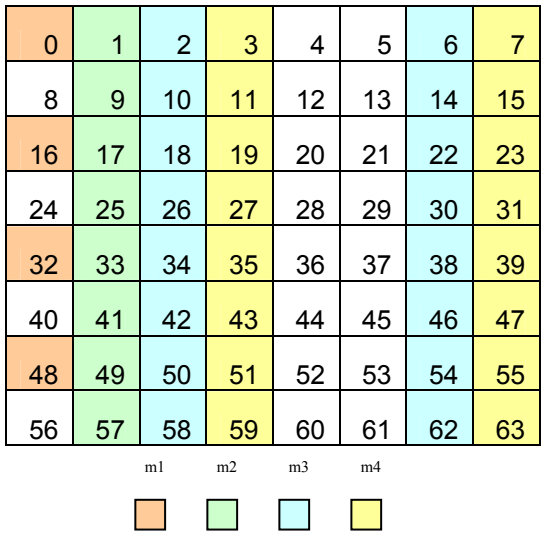

Figure 5-3: Scheduling map of a static slot for the case, allocation with minimum jitter

In some experiments jitter is allowed to minimize the SA and U_{ID}, and to increase the utilization. In these cases, messages with the possibility of sharing the same slot are assigned to one slot without jitter considerations. Again m_{RT} values are changed to new f_{RT} values because of the reasons given in previous discussion. Possible largest value that is less than the m_{RT} or guarantees on time transmission is set for f_{RT} value. For example if the m_{RT} is 50, then the largest possible f_{RT} value that is a power of two and less than 50 is 32, and for the m_{RT} values 100, 250 and 400 f_{RT} is set to be 64. If we consider the previous example, new f_{RT} values for the case that jitters are allowed are 64, 64, 64, and 16 respectively. Then SA for this case is

$$SA = 1/64 + 1/64 + 1/64 + 1/16 \approx 0.11 < 1 \qquad (4.3)$$

As can be seen, *SA* is reduced for the same message set and therefore Utilization (*U*) is increased with the cost of jitter. Then the resulting message scheduling is shown in the Figure 5-4.

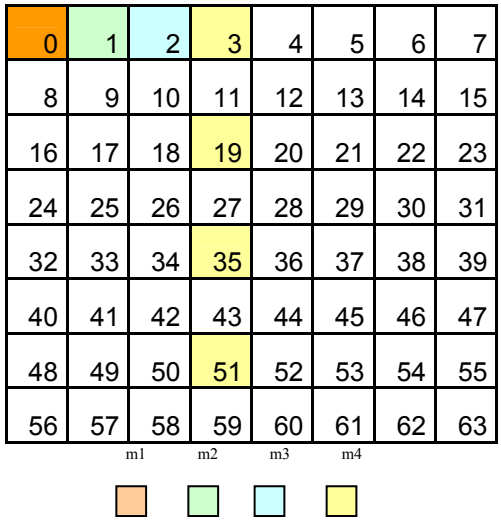

Figure 5-4: Scheduling map of a static slot for the case, minimum *SA* (Jitter is allowed)

Dynamic slot allocation of sporadic messages can not be done in the same way as periodic messages since the generation times of sporadic messages can not be known prior to execution. For this reason sporadic messages are assigned to dynamic slots based on their priorities (Message ID). The message with the highest priority is assigned to the first dynamic slot. In our experiments the message with the lowest message ID has the highest priority. These dynamic slots consist of minislots. If the message assigned to dynamic slot is not ready then the duration of dynamic slot is equal to one minislot. If message transmission takes place then the duration of the dynamic slot is equal to the message size plus the protocol overheads.

When message scheduling is completed the needed numbers of static and dynamic slots are identified. By taking into account this information, tables that include cluster wide configuration parameters are constructed in this step. First table

consists of *message specific configuration parameters* and is given in Table 5-2. Second table consists of general parameters that are common to all nodes and called as the *cluster configuration parameters* and given in Table 5-3.

Table 5-2: Message Specific Configuration Parameters

Message ID	**Segment**	**Static/ Dynamic Slot**	**Repetition Time (f_{RT})**	**Offset (for periodic messages)**
Message Identification	Static/ Dynamic	Slot ID	Repetition Time	Offset value for cycle multiplexing

Table 5-3: Cluster Configuration Parameters

	Configuration Parameter	**Message Set Feature**	**Note**
Static Segment	gdCycle (μs)	Message Periods	Greatest Common Divisor of message periods.
	gPayloadLengthStatic (2-Byte-Words)	Message Sizes	Minimum possible payload length is maximum message size in the set
	gdActionPointOffset (MT)	NA	It is used as a safety margin
	gdStaticSlot (MT)	Message Sizes	Static Slot Size is the sum of payload length, action point offset and protocol overheads
	gNumberOfStaticSlots	Number of Messages	According to the scheduling scheme number of static slots can be configured
	Static Segment Duration	Number of Messages and Message Sizes	Static segment length is equal to the Static Slot size multiplied with Number of slots. Static segment length is smaller than Cycle Time

Table 5-3 (continued)

Dynamic Segment	gNumberOfMiniSlots	Maximum size and minimum deadline of Sporadic Messages	While determining the number of minislots, minislot duration and timing characteristics of sporadic messages should be considered
	gdMinislot (MT)	Timing characteristics of sporadic messages	
	gdMiniSlotActionPointOffset (MT)	Timing characteristics of sporadic messages	
	gdDynamicSlotIdlePhase (Minislot)	Timing characteristics of sporadic messages	
	Dynamic Segment Duration	Timing characteristics of sporadic messages	
Cycle	gdNIT (MT)		Remaining time from Static and Dynamic segments
	gdMacrotick (μs)		

The configuration parameters are set as follows:

gdCycle (μs): This value (cycle time) is chosen according to the message periods. As mentioned in [21], choosing greatest common divisor of message periods as cycle length makes it possible to arrange scheduling in an optimized way and transmission with zero jitter.

gPayloadLengthStatic: Then by considering the message sizes in the static segment, Payload Length (gPayloadLengthStatic) of static slots is decided. In FlexRay static segment all slots have to be in the same length. This leads to payload length to be chosen as the maximum message size to cover all nodes and messages. Actually payload length can be chosen a greater value for future reservations. However to increase the utilization for the given message set, it is

chosen as the possible smallest value. In addition to the payload length, FlexRay protocol adds some parameters to convert the payload to the FlexRay frame. These parameters are 5 bytes of Header and 3 bytes of Trailer. Furthermore physical transmission of FlexRay frame requires additional protocol overheads. These are 1 bit of Frame Start Sequence (gdFSS), 2 bits of Frame End Sequence (gdFES), 3-15 bits (at 10 Mbps baud rate) of Transmission Start Sequence (gdTSSTransmitter), 11 bits of Communication Idle Delimiter (cChannelIdleDelimiter), and 2 bits of Byte Start Sequence (gdBSS) for every byte of FlexRay frame. Then total FlexRay frame with physical transmission requirements is given as

$$S_{PF}[\text{bit}] = S_F[\text{byte}]*10 + gdTSSTransmitter + cChannelIdleDelimiter + gdFSS + gdFES \quad (4.4)$$

Where S_{PF} is the physical FlexRay Frame Size in terms of number of bits and S_F is the FlexRay Frame in terms of number of bytes [13]. The picture of the physical frame is given in Figure 5-5.

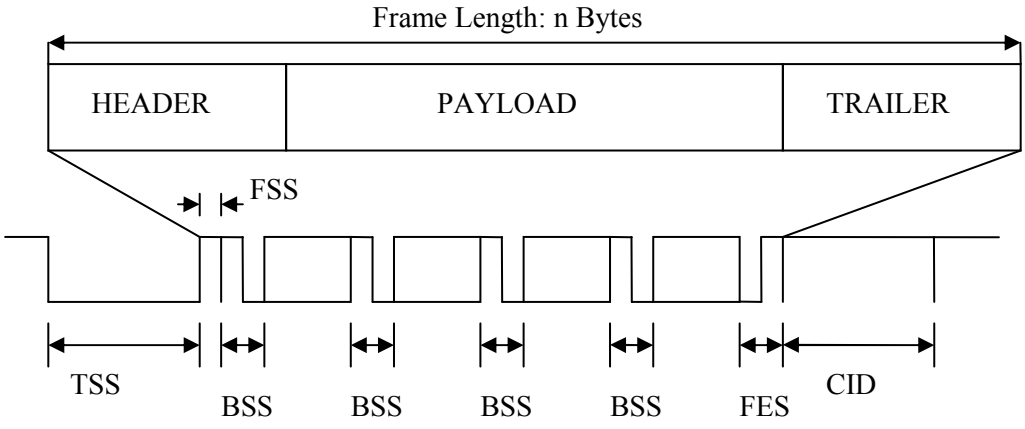

Figure 5-5: Physical FlexRay Frame [22]

gdActionPointOffset : In the static slots there is a point where the communication starts and ends. This point called as static slot Action Point Offset (gdActionPointOffset). In the configuration this value is defined in terms of number of Macroticks (MT). Communication starts gdActionPointOffset MT after the slot boundary and ends gdActionPointOffset MT before the next slot boundary.

The reason of action point offset is to prevent the messages to violate the slot boundaries during transmission.

gdStaticSlot (MT): It is the static slot size. In addition to the Physical Frame Size and Action Point Offset, other protocol parameters also add up to static slot size. These parameters are Minimum Propagation Delay (gdMinPropagationDelay), Maximum Propagation Delay (gdMaxPropagationDelay), MT duration (gdMacrotic), and Maximum Clock Deviation (cClockDeviationMax) as defined in [13].

gNumberOfStaticSlots: After deciding these four parameters, number of Static Slots (gNumberOfStaticSlots) is defined. It is the summation of the allocated slots for all nodes plus future reservations for network expansions.

Other than payload length, the parameters used in the configuration of physical static frame length and static slot size are independent from the message set. Static Slot Size times the number of Static Slots gives static segment duration in terms of number of MTs.

The above mentioned parameters are for the static segment. For the dynamic segment additional parameters need to be defined. These are Minislot Duration (gdMinislot) in terms of number of MTs, Minislot Action Point Offset (gdMinislotActionPointOffset) in terms of number of MTs, Dynamic Slot Idle Phase (gdDynamicSlotIdlePhase) in terms of number of Minislots and total Number of Minislots (gNumberOfMiniSlots). Transmission starts at minislot action point offset MT from the minislot boundary. Unlike the static segment, communication stops at slot boundary.

Then dynamic segment size given as

$$S_{DS} = gdMinislot * gNumberOfMinislots \tag{4.5}$$

Where, S_{DS} is the size of dynamic segment. Dynamic Slot Idle Phase is the conclusion of the dynamic segment and no transmission takes place in this phase.

Every communication cycle ends up with Network Idle Time (gdNIT). While determining the static segment and dynamic segment sizes it should be noticed that their total duration can not exceed Cycle Time - gdNIT. Therefore gdNIT also need to be determined cluster wide together with other parameters.

The general picture after setting all the mentioned parameters is given in Figure 5-6. In this figure header and trailer segments have fixed size and all the other parameters are configurable.

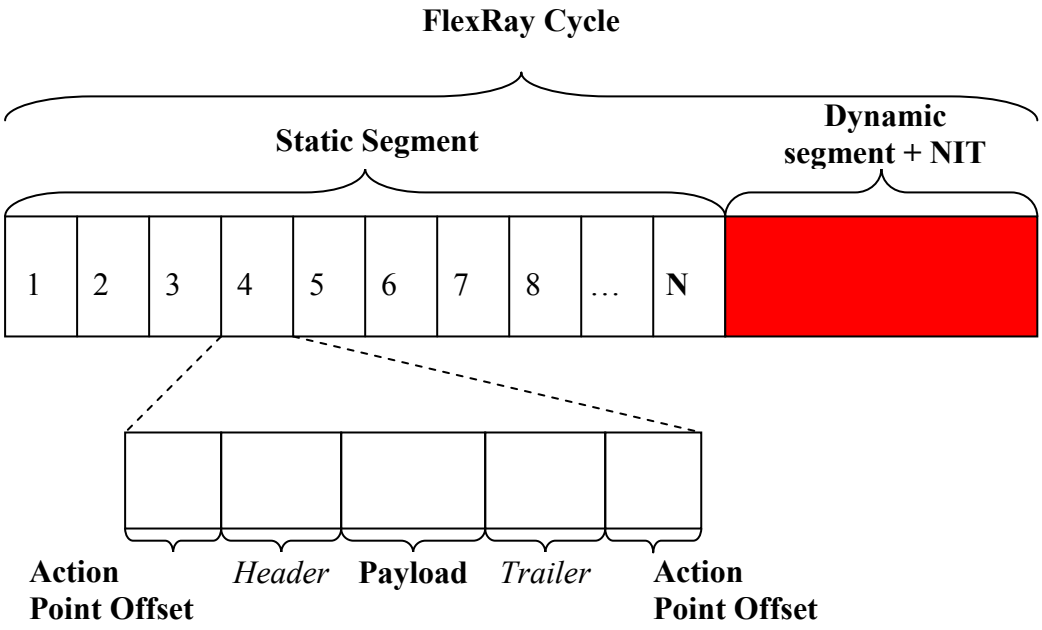

Figure 5-6: Configuration of Static Segment

5.1.3 Implementation of Configuration Parameters

After the scheduling scheme is constructed and tables for the parameters are created, scheduling is implemented by using a universal FlexRay configuration tool which is FlexConfig™ Developer. By considering the scheduling table, configuration parameters are set to related values. When implementation is done configuration files (*.chi) is generated for each node by the FlexConfig. These files include the related register values for the communication controller of the

Evaluation Boards to perform the intended application. Communication Controller configuration is realized by including FlexConfig output files to the main code of the microcontroller. In the start up of the FlexRay, this configuration values are loaded to the registers of the Communication Controller by the microcontroller.

5.1.4 Microcontroller Implementation

The next step after configuration is implemented is the preparation of the related microcontroller codes for each node. In the code needed number of periodic and sporadic messages is created by timer interrupts and is sent to the preconfigured buffers of nodes. Then Communication Controller transmits messages according to configuration parameters and protocol specifications.

5.1.5 Analysis of the Results

Constructed FlexRay network and designed experiment are monitored with FlexRay bus analyzer tool. The performance metrics, defined in Section 3.2, are calculated from data that is logged by using the analyzer and then analyzed.

CHAPTER 6

EXPERIMENTS

The evaluation of the FlexRay network is performed by a number of experiments. The purpose, method, and analysis of the designed experiments are presented in this section. Experiments are separated into two parts as Static Segment Experiments (Section 0) and Dynamic Segment Experiments (Section 6.2). FlexRay static and dynamic segments are two separate parts of the protocol. Their structure and timing hierarchy are different. Structure of the FlexRay allows internal configuration of static and dynamic segments to be done independently. The only effect to each other, after cycle duration, symbol window and NIT are fixed, is the length of each segment. Sum of the durations of two segments can not exceed the remaining time (Cycle duration - Symbol Window Duration - NIT) and also static segment can be considered as the time offset before the dynamic segment. Therefore experiments are designed, performed and analyzed separately.

Manipulation and control of protocol variables are easy to handle in a relatively small network before studying a larger scale network. Therefore all the related FlexRay protocol variables in the experiments are investigated to see if the protocol works as expected on a 3 node network. In addition test setup is verified. After the verification, number of nodes is increased to 6 and similar experiments are performed on the new network configuration. This approach reduced the time for finding and fixing the errors, constructing the stable FlexConfig files, and getting the bug free microcontroller codes. An illustration of the 3 node network is shown in Figure 6-1 and a photograph of the 6 node network is presented in Figure 6-2.

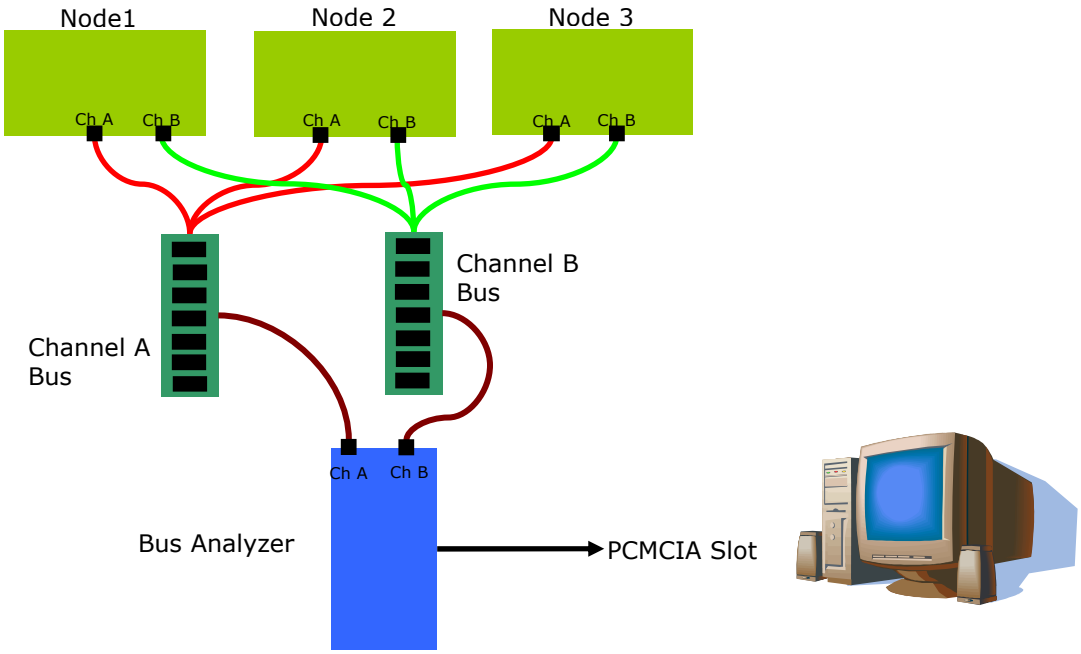

Figure 6-1: 3 Node Network Setup

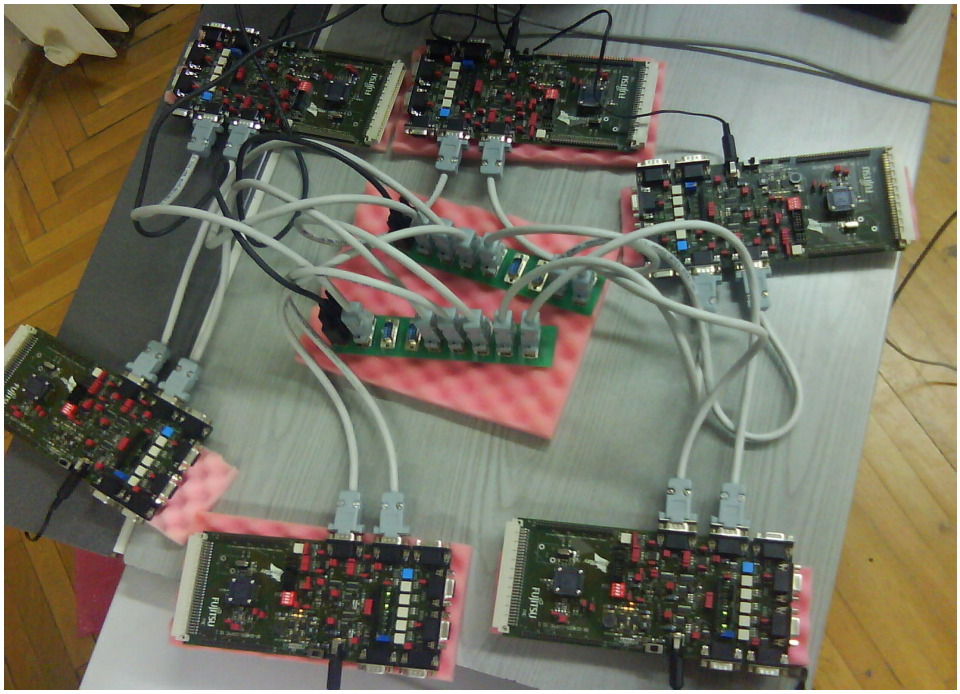

Figure 6-2: 6 Node Network

6.1 Static Segment Experiments

Static segment experiments are designed to see the effect of scheduling methods on the performance metrics, Used Static Slot ID (U_{ID}), Static Slot Allocation (SA), Allocated Bandwidth (BW_A), and Utilization (U) given in Section 3.2. First 5 experiments are performed on the 3-node network and remaining 5 experiments are performed on the 6-node network. In the first experiment the simplest scheduling method, which is one static slot is fully allocated to each message, is used. Then in the experiments 2, 3, and 4, the scheduling method given in Section 3.4 is applied which resulted in improving the performance metrics namely increasing of utilization (U) and reducing slot allocation (SA) and static slot ID usage (U_{ID}). In the experiment 5 signal packing is applied to further increase the utilization. While improving these metrics delay and jitter are kept within acceptable levels. The trade off between the Utilization/slot ID usage/Slot Allocation and Jitter/Delay are illustrated in our results. For the delay and jitter we looked at the maximum and average values. Since, exceeding the deadline for any message is unacceptable, maximum delay for every Slot ID is investigated. Average delay and jitter values are measured to decide if the used method is providing a deterministic trend for the message transmissions. Experiments 6 and 7 repeat the previous experiments 3 and 4 repeated on the 6 node network. Cycle duration is reduced to 2.5 ms in the experiments 8 and 9 to decrease delay values. While messages are generated synchronously with scheduling offsets in experiment 8, in experiment 9 the first occurrences of all messages are set to be the same cycle. Signal packing is applied in the experiment 10 to further improve the utilization, U_{ID}, and SA.

The offset and repetition time values in the communication controller files (*.mhx), and Slot ID's of the messages are configured to achieve the desired schedule. The values of these variables for each message are given in the related experiments. Other configuration variables are not affected to apply these methods therefore they are fixed. These fixed configuration variables are given in Table 6-1 to Table 6-3 with the related experiment numbers.

Table 6-1: Fixed Configuration Parameters – Static Segment (Experiments 1, 2, 3, 4, 6, 7)

Configuration Parameter Name	Value
gdCycle (µs)	5000 µs
gPayloadLengthStatic (2-Byte-Words)	5 (10 Bytes)
gdActionPointOffset (MT)	5 MT
gdStaticSlot (MT)	31 MT
gNumberOfStaticSlots	64
Static Segment Length	64*31 = 1984 MT
gdMacrotick (µs)	1 µs

Table 6-2: Fixed Configuration Parameters – Static Segment (Experiments 5, 10)

Configuration Parameter Name	Value
gdCycle (µs)	5000 µs
gPayloadLengthStatic (2-Byte-Words)	12 (Experiment 5), 4 (Experiments 10)
gdActionPointOffset (MT)	5 MT
gdStaticSlot (MT)	45 MT (Experiment 5), 29 MT (Experiment 10)
gNumberOfStaticSlots	64
Static Segment Length	2880 MT (Experiment 5), 1856 (Experiment 10)
gdMacrotick (µs)	1 µs

Table 6-3: Fixed Configuration Parameters – Static Segment (Experiments 8, 9)

Configuration Parameter Name	Value
gdCycle (µs)	2500 µs
gPayloadLengthStatic (2-Byte-Words)	5 (10 Bytes)
gdActionPointOffset (MT)	5 MT
gdStaticSlot (MT)	31 MT
gNumberOfStaticSlots	32
Static Segment Length	32*31 = 992 MT
gdMacrotick (µs)	1 µs

The message set used in the experiments is shown in Table 6-4. This message set is taken from TOFAŞ that means it is used in real automobiles. The first 41 messages are used in the 3 node experiments and all of the 51 messages are used in the 6 node experiments. Minimum period value for the messages is 5 ms and all periods are integer multiples of 5 ms. Therefore gdCycle (Cycle duration) is chosen as 5 ms as discussed in Section 5.1.2. Some timing data are packed inside the message data and sent over the network to collect message delay and jitter statistics. These data are the cycle counter value and macrotick counter value when the message is created by the node that the message belongs to, message ID, and period of the message. Because of this data, gPayloadLengthStatic (Payload length of the static segment frames) is set to be 10 Bytes for the experiments 1, 2, 3, 4, 6, 7, 8, and 9. In experiments 5 and 10 actual message sizes are used. Action point offset in static slots is set to be 5 MT. Macrotick duration is set to be 1μs which is 10 bit duration in a 10 Mbps cluster. With these values minimum length of a Static Slot (gdStaticSlot) is calculated by the FlexConfig tool for each experiment. A total of 64 static slots are configured for the static segment to reserve free slots for future extensions. In this way, the addition of new messages does not require a configuration change and does not affect the rest of the network.

Messages are created at the beginning of each FlexRay cycle by software interrupts of the MCUs of the evaluation boards. The periods of these interrupts are the same as the period of the related messages. When the software interrupt occurs, MCU checks the macrotick counter value of the Communication Controller (CC) and according the counter value it adjusts the new interrupt time so that messages are created at the beginning of each FlexRay cycle and the generation time is synchronized with beginning of FlexRay cycles. Since MCU and CC have different clocks there is a rate difference and this difference is suppressed by the above mentioned software coded control algorithm. This algorithm guarantees that interrupts occur between the 50^{th} and 150^{th} Macrotick counter values. Then the related message is sent to the reserved buffer (this transfer also consumes at most 30 MT) of the CC and wait for transmission. Transmission takes place according to

the f_{RT} and *offset* values. The duration of a static slot is 31 MT. If the generation time of the message is 150th MT and considering the transfer to the buffer (the sum is around 180 MT), the value of the current transmission slot can be 6 (6 * 31 MT = 186 MT) when the message is ready to be transmitted. Therefore first Static Slot ID used for the message scheduling is 7 to guarantee that no message will miss its static slot beginning time at the generation cycle. During the experiments, logs are recorded.

Table 6-4: Message set used in the Static Segment Experiments

Message ID	Message Type	Message Period	Message Size	Sender Node	Receiver Node
Message 1	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 2	Periodic	5 ms	8 bytes	Node 2	Analyzer
Message 3	Periodic	20 ms	8 bytes	Node 2	Analyzer
Message 4	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 5	Periodic	10 ms	8 bytes	Node 1	Analyzer
Message 6	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 7	Periodic	10 ms	8 bytes	Node 1	Analyzer
Message 8	Periodic	10 ms	8 bytes	Node 1	Analyzer
Message 9	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 10	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 11	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 12	Periodic	20 ms	8 bytes	Node 2	Analyzer
Message 13	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 14	Periodic	20 ms	8 bytes	Node 2	Analyzer
Message 15	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 16	Periodic	10 ms	8 bytes	Node 1	Analyzer
Message 17	Periodic	10 ms	8 bytes	Node 1	Analyzer
Message 18	Periodic	10 ms	4 bytes	Node 1	Analyzer
Message 19	Periodic	100 ms	8 bytes	Node 1	Analyzer
Message 20	Periodic	50 ms	8 bytes	Node 1	Analyzer
Message 21	Periodic	100 ms	2 bytes	Node 2	Analyzer
Message 22	Periodic	100 ms	8 bytes	Node 2	Analyzer
Message 23	Periodic	100 ms	8 bytes	Node 2	Analyzer
Message 24	Periodic	250 ms	2 bytes	Node 3	Analyzer
Message 25	Periodic	500 ms	1 bytes	Node 3	Analyzer
Message 26	Periodic	250 ms	3 bytes	Node 1	Analyzer
Message 27	Periodic	10 ms	8 bytes	Node 2	Analyzer
Message 28	Periodic	100 ms	4 bytes	Node 1	Analyzer
Message 29	Periodic	100 ms	4 bytes	Node 1	Analyzer
Message 30	Periodic	100 ms	4 bytes	Node 1	Analyzer
Message 31	Periodic	2000 ms	8 bytes	Node 1	Analyzer
Message 32	Periodic	2000 ms	8 bytes	Node 1	Analyzer
Message 33	Periodic	1000 ms	6 bytes	Node 1	Analyzer
Message 34	Periodic	1000 ms	2 bytes	Node 1	Analyzer

Table 6-4 (continued)

Message 35	Periodic	20 ms	8 bytes	Node 3	Analyzer
Message 36	Periodic	2000 ms	8 bytes	Node 1	Analyzer
Message 37	Periodic	2000 ms	8 bytes	Node 1	Analyzer
Message 38	Periodic	2000 ms	8 bytes	Node 2	Analyzer
Message 39	Periodic	2000 ms	8 bytes	Node 2	Analyzer
Message 40	Periodic	2000 ms	8 bytes	Node 3	Analyzer
Message 41	Periodic	100 ms	8 bytes	Node 1	Analyzer
Message 42	Periodic	20 ms	2 bytes	Node 4	Analyzer
Message 43	Periodic	500 ms	1 bytes	Node 4	Analyzer
Message 44	Periodic	100 ms	2 bytes	Node 4	Analyzer
Message 45	Periodic	250 ms	8 bytes	Node 4	Analyzer
Message 46	Periodic	10 ms	4 bytes	Node 4	Analyzer
Message 47	Periodic	500 ms	8 bytes	Node 5	Analyzer
Message 48	Periodic	250 ms	8 bytes	Node 5	Analyzer
Message 49	Periodic	10 ms	4 bytes	Node 5	Analyzer
Message 50	Periodic	500 ms	4 bytes	Node 6	Analyzer
Message 51	Periodic	500 ms	4 bytes	Node 6	Analyzer

6.1.1 Experiment 1: 3 Nodes, One Message per Static Slot

In this experiment one static slot is allocated for each message in every cycle. The aim of this experiment is to investigate the values of the performance metrics for this simplest scheduling policy. Then these values for the metric will be improved for the network in the next experiments with different scheduling approaches. The table that shows the values of the message specific configuration parameters is given in Table 6-5. Duration of the experiment is approximately 60 seconds. An example view of the static slots in terms of allocation in cycles 0 to 63 is given in Figure 6-3.

Table 6-5: Experiment 1: Message Specific Configuration Parameters

Message ID	Segment	Slot ID	Repetition	Offset
Message 1	Static	44	1	0
Message 2	Static	45	1	0
Message 3	Static	42	1	0
Message 4	Static	46	1	0
Message 5	Static	47	1	0
Message 6	Static	43	1	0
Message 7	Static	7	1	0

Table 6-5 (continued)

Message 8	Static	8	1	0
Message 9	Static	9	1	0
Message 10	Static	10	1	0
Message 11	Static	11	1	0
Message 12	Static	12	1	0
Message 13	Static	13	1	0
Message 14	Static	14	1	0
Message 15	Static	15	1	0
Message 16	Static	16	1	0
Message 17	Static	17	1	0
Message 18	Static	18	1	0
Message 19	Static	19	1	0
Message 20	Static	20	1	0
Message 21	Static	21	1	0
Message 22	Static	22	1	0
Message 23	Static	23	1	0
Message 24	Static	24	1	0
Message 25	Static	25	1	0
Message 26	Static	26	1	0
Message 27	Static	27	1	0
Message 28	Static	28	1	0
Message 29	Static	29	1	0
Message 30	Static	30	1	0
Message 31	Static	31	1	0
Message 32	Static	32	1	0
Message 33	Static	33	1	0
Message 34	Static	34	1	0
Message 35	Static	35	1	0
Message 36	Static	36	1	0
Message 37	Static	37	1	0
Message 38	Static	38	1	0
Message 39	Static	39	1	0
Message 40	Static	40	1	0
Message 41	Static	41	1	0

Total of 41 static slots are allocated in each cycle for 41 messages. Repetition times of all messages are selected equal to 1 to reserve the respective static slots in every cycle whether or not a message is ready to be transmitted. If there is a message ready to be transmitted then allocated frame is sent with the message, if there is no message ready to be transmitted then the frame is sent empty. Then U_{ID} and allocated number of static slots have the equal value,

$$U_{ID} = SA = 41 \tag{5.1}$$

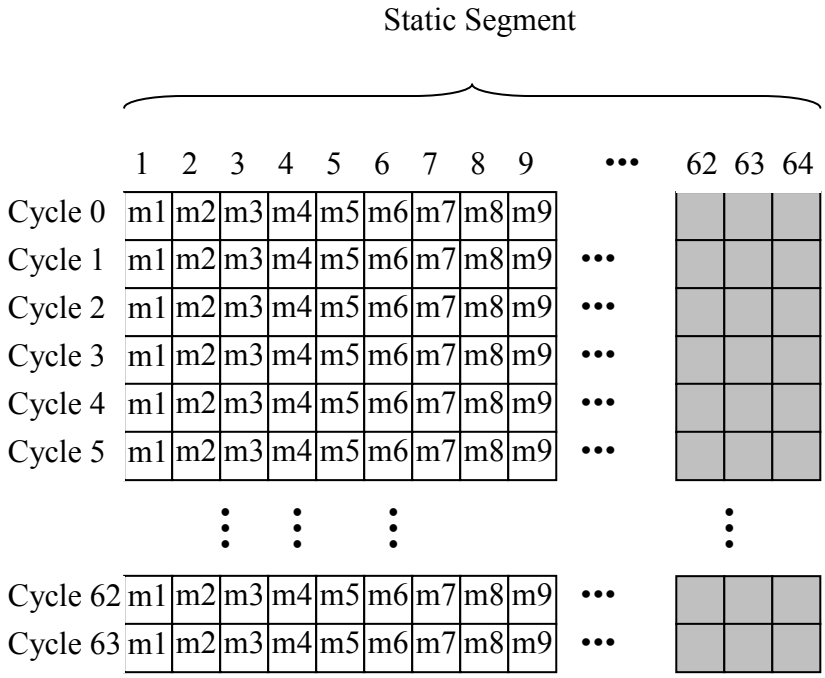

Figure 6-3: Experiment 1, Static slot allocation, one slot is allocated for each message in every cycle

Duration of one static slot is 31 MT and MT duration is 1 µs. Duration of one cycle is equal to 5000 µs. Then Allocated portion of the cycle is calculated as

$$BW_A = (41*31)/5000 = 0.2542 = 25.42\% \quad (5.2)$$

and this value shows that 25.42% of the 10 Mbps bandwidth is allocated for the current message transmission.

Rate of the transmitted messages is measured as B_U = 2016 messages per second during the experiment.

In every cycle 41 messages are allowed to be transmitted. Then allocated number of static slots per one second is given as

$$B_A = (1/0.005)*41 = 8200 \quad (5.3)$$

Then Utilization is given as

$$U = \frac{B_U}{B_A} = 2016/8200 = 0.246 = 24.6\% \quad (5.4)$$

The available bandwidth of the FlexRay network is 10 Mbps. 25.42% of available bandwidth is allocated for the current message transmission and 24.6% of this allocated bandwidth is actually used. Then used bandwidth is given as

$$BW_U = 10 \text{ Mbps} * 0.2542 * 0.246 \approx 624{,}960 \text{ bps} \quad (5.5)$$

Used bandwidth shows the traffic load of transmitted messages on the FlexRay network.

The results of the delay and jitter calculations for each message ID are given in Appendix I Table A-1. A figure that shows the maximum and mean values of delays and jitters with respect to static slot IDs are given in Figure 6-4. Since the values of average and maximum jitter are very low as compared to other values for some messages, they may not be seen for this scale. This situation is also valid in other static segment experiments.

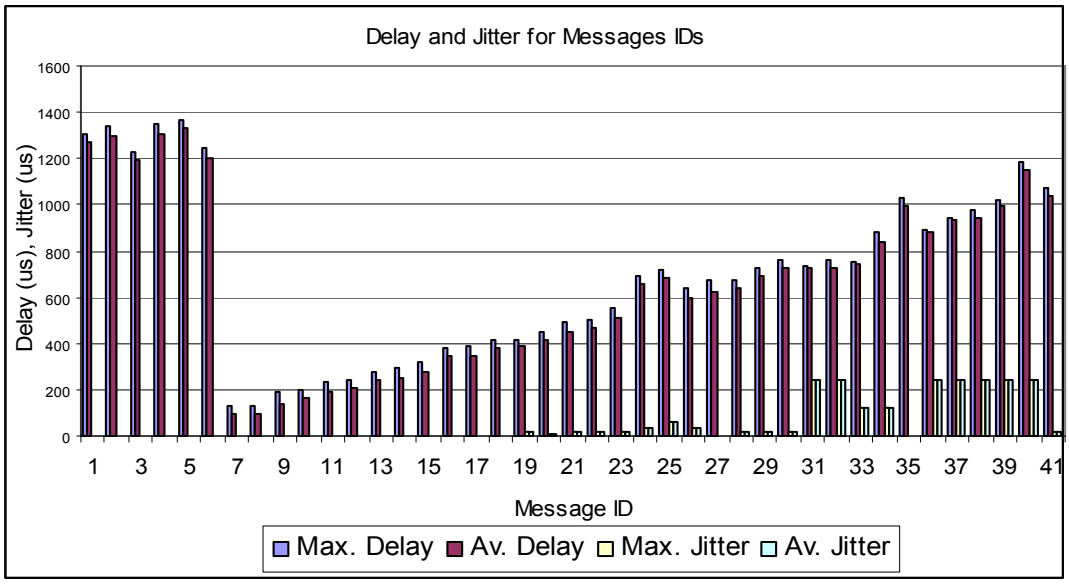

Figure 6-4: Experiment 1 – delay (µsec) & jitter (µsec) vs. Message ID

Maximum and average delay values for the message IDs are almost equal and quite shorter than 5 ms (*gdCycle*). Since the messages are created at the beginning of FlexRay cycles before their allocated static slots and slots allocated in every cycle, none of the messages experiencing a delay greater than cycle duration. If the allocated static slot IDs and delay values are considered, it can be seen that as the number of static slot ID increases delay values are also increasing. Messages with lower slot IDs experiencing lower delays. The jitter is almost zero for the messages with small periods. As the period values are increasing jitter is also increasing. The jitter values are expected to be zero because periods of the messages are integer multiples of cycle duration and static slots is allocated also periodically. However jitter is more than zero, because of the clock drifts resulted in the local oscillators of evaluation boards. This drift is approximately 3 μs in every 2 FlexRay cycles. Message IDs with higher periods are experiencing more jitter, because there are more cycles between consecutive messages.

6.1.2 Experiment 2: 3 Nodes, Cycle Filtering without Jitter

In this experiment one slot is allocated for each message as in Experiment 1. However in this case the method of cycle filtering is used hence the messages are sent periodically only in the cycles selected according to the message periods. This method reduces the *SA* for the network. An example view of the resulting slot allocation is given in Figure 6-5. In the figure the numbered slots show that in that cycle the given message is allowed to send. If the given message is not ready at the time of allocated slot, slot will be empty. No other messages can be transmitted. Gray filled slots means that they are not allocated to any messages. No transmissions take place in the slot at the gray colored cycles. They are free slots to be considered for the further message allocation by the same node.

Cycle filtering is performed by the Communication Controller in run time. When the cycle counter value matches with f_{RT} and *offset* value it permits for transmission. The cycle filtered scheduling is applied to increase utilization as much as possible.

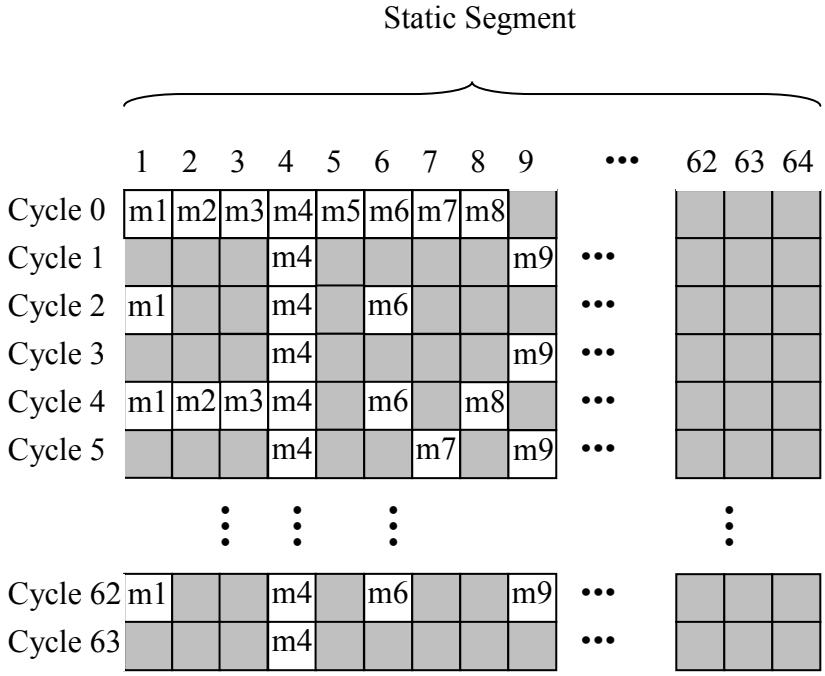

Figure 6-5: Experiment 2, Static slot allocation, cycle filtering is used

Duration of the experiment is approximately 60 seconds. The table that shows the values of the message specific configuration parameters is given in Table 6-6.

Table 6-6: Experiment 2: Message Specific Configuration Parameters

Message ID	Segment	Slot ID	Repetition	Offset
Message 1	Static	44	2	0
Message 2	Static	45	1	0
Message 3	Static	42	4	0
Message 4	Static	46	2	0
Message 5	Static	47	2	0
Message 6	Static	43	2	0
Message 7	Static	7	2	0
Message 8	Static	8	2	0
Message 9	Static	9	2	0
Message 10	Static	10	2	0
Message 11	Static	11	2	0
Message 12	Static	12	4	0
Message 13	Static	13	2	0
Message 14	Static	14	4	0
Message 15	Static	15	2	0

Table 6-5 (continued)

Message 16	Static	16	2	0
Message 17	Static	17	2	0
Message 18	Static	18	2	0
Message 19	Static	19	4	0
Message 20	Static	20	2	0
Message 21	Static	21	4	0
Message 22	Static	22	4	0
Message 23	Static	23	4	0
Message 24	Static	24	2	0
Message 25	Static	25	4	0
Message 26	Static	26	2	0
Message 27	Static	27	2	0
Message 28	Static	28	4	0
Message 29	Static	29	4	0
Message 30	Static	30	4	0
Message 31	Static	31	16	0
Message 32	Static	32	16	0
Message 33	Static	33	8	0
Message 34	Static	34	8	0
Message 35	Static	35	4	0
Message 36	Static	36	16	0
Message 37	Static	37	16	0
Message 38	Static	38	16	0
Message 39	Static	39	16	0
Message 40	Static	40	16	0
Message 41	Static	41	4	0

The number of static slots that is used is again

$$U_{ID} = 41 \tag{5.6}$$

However in this experiment slots are not allocated to the messages in every cycle. Instead allocation is done according to repetition times. Then *SA* for this configuration is calculated as

$$SA = \frac{1}{1} + 18 * \frac{1}{2} + 13 * \frac{1}{4} + 2 * \frac{1}{8} + 7 * \frac{1}{16} = 13.9375 \tag{5.7}$$

Compared to the *SA* value for Experiment 1, there is a significant improvement in this scheduling. Allocated bandwidth is

$$BW_A = (13.9375*31)/5000 = 0.0864 = 8.64\% \quad (5.8)$$

Allocated number of static slots per one second is

$$B_A = 200*13.9375 = 2787.5 \quad (5.9)$$

Total number of messages transmitted during the experiment is 125630. This corresponds to $B_U = 2016$ messages per second. Then Utilization is computed as

$$U = 2016/2787.5 = 72\% \quad (5.10)$$

Used bandwidth is again, 624,960 bps

$$BW_U = 10 \text{ Mbps} * 0.72 * 0.0864 \approx 624{,}960 \text{ bps} \quad (5.11)$$

Delay and jitter values for each message is given in Appendix I Table A-2. A figure that shows the maximum and mean values of delays and jitters with respect to static slot IDs are given in Figure 6-6. The maximum and average delay values for the scheduling methods that ensure minimum jitter are very close to each others because of the deterministic structure of the static segment. Actual values are given in the Appendix section for a better judgment.

The jitter values for the Message IDs are equal to the values in Experiment 1. This result shows the correctness of the method (message scheduling without jitter) given in Section 5.1.2. Although maximum and average delay values are approximately the same, the trend of the delay values show an important change as compared to Experiment 1. If the slot is not allocated for the message at the generation cycle, then transmission has to wait until first cycle that the message is allowed (allocated) to be sent. This delay depends on the starting state of the network. Since the messages and FlexRay cycles are periodic after starting delays follow the same pattern. The maximum delay that can occur is less than (f_{RT} – *1)*gdCycle* + some offset time. The reason for this time offset is given in Section 6.1.

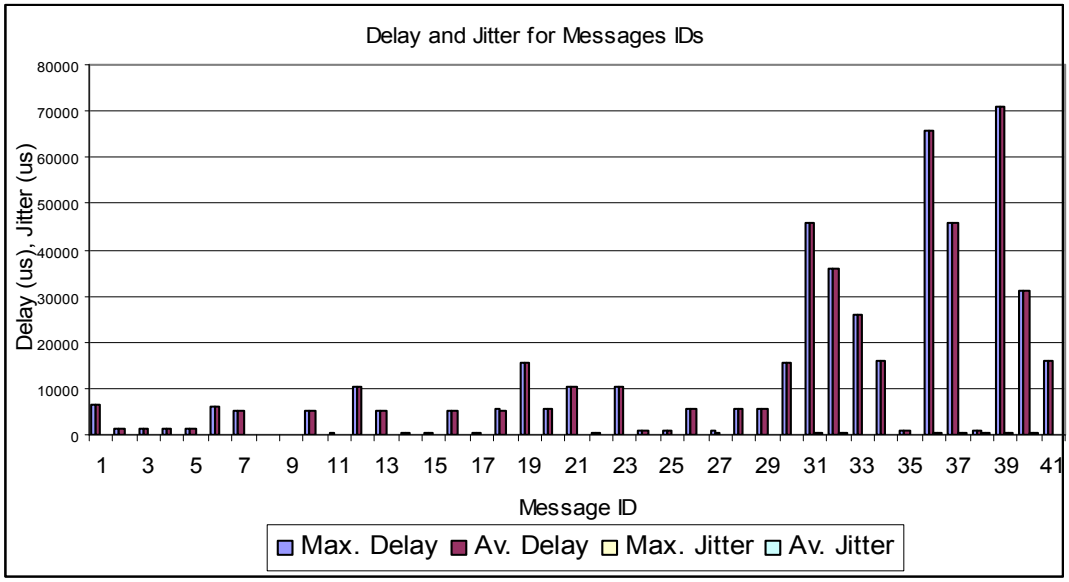

Figure 6-6: Experiment 2 – delay (µsec) & jitter (µsec) vs. Message ID

6.1.3 Experiment 3: 3 Nodes, Cycle Multiplexing without Jitter

In this experiment more than one message are allocated to one static slot whenever possible. In the experiment 2, cycle filtering was used and then some slots had empty cycles. These empty cycles are now allocated to other messages of the same node by applying the cycle multiplexing. Assignment of the messages to the static slots is performed according to the method defined in [21]. By this way allocated number of static slots ID is decreased and extra free slot IDs are made available for the possible addition of new nodes to the cluster. As can be seen from the Table 6-7 the allocated static slot ID is reduced from 41 slots where:

$$U_{ID} = 16 \tag{5.12}$$

A figure that shows the cycle multiplexed allocation is given in Figure 6-7.

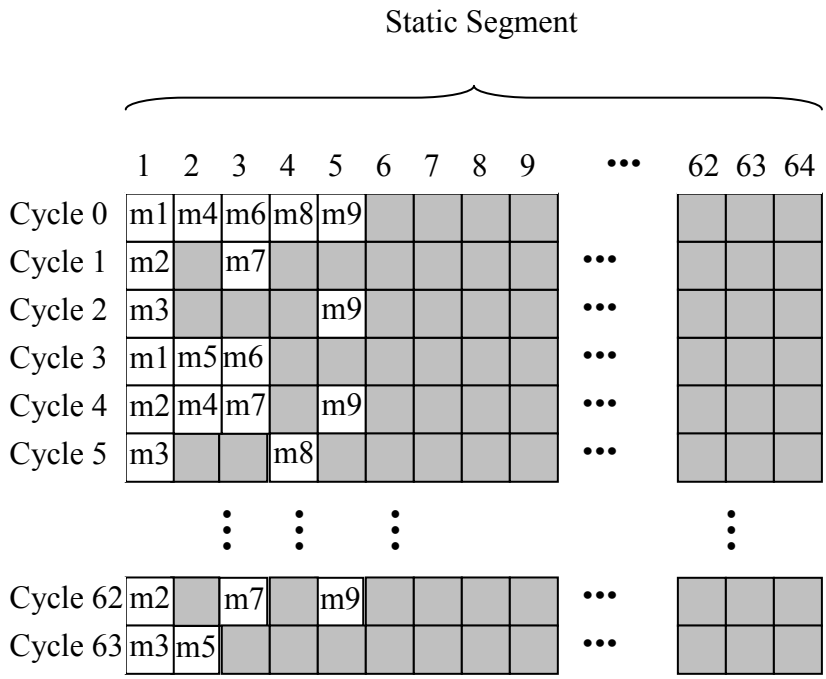

Figure 6-7: Experiment 3, Static slot allocation, message filtering and cycle multiplexing is used

Table 6-7: Experiment 3: Message Specific Configuration Parameters

Message ID	Segment	Static Slot	Repetition	Offset
Message 1	Static	14	2	0
Message 2	Static	13	1	0
Message 3	Static	18	4	0
Message 4	Static	14	2	1
Message 5	Static	7	2	0
Message 6	Static	15	2	0
Message 7	Static	7	2	1
Message 8	Static	8	2	0
Message 9	Static	15	2	1
Message 10	Static	16	2	0
Message 11	Static	16	2	1
Message 12	Static	19	4	0
Message 13	Static	17	2	0
Message 14	Static	18	4	2
Message 15	Static	17	2	1
Message 16	Static	8	2	1
Message 17	Static	9	2	0
Message 18	Static	9	2	1
Message 19	Static	11	4	0
Message 20	Static	10	2	0
Message 21	Static	19	4	1

Table 6-7 (continued)

Message 22	Static	19	4	2
Message 23	Static	19	4	3
Message 24	Static	21	2	0
Message 25	Static	22	4	0
Message 26	Static	10	2	1
Message 27	Static	18	2	1
Message 28	Static	11	4	1
Message 29	Static	11	4	2
Message 30	Static	11	4	3
Message 31	Static	12	16	1
Message 32	Static	12	16	2
Message 33	Static	12	8	3
Message 34	Static	12	8	5
Message 35	Static	22	4	1
Message 36	Static	12	16	6
Message 37	Static	12	16	7
Message 38	Static	20	16	0
Message 39	Static	20	16	2
Message 40	Static	22	16	2
Message 41	Static	12	4	0

Allocated number of static slots is:

$$SA = 13.9375 \tag{5.13}$$

This value is the same as in experiment 2 because same repetition times are used for each frame that message is transmitted in.

Allocated bandwidth is

$$BW_A = (13.9375*31)/5000 = 0.0864 = 8.64\% \tag{5.14}$$

Allocated number of static slot per one second is

$$B_A = 200*13.9375 = 2787.5 \tag{5.15}$$

Total number of messages transmitted during the experiment is 121824. This corresponds to $B_U = 2016$ messages per second. Then Utilization is given as

$$U = 2016/2787.5 = 72\% \tag{5.16}$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.72 * 0.0864 \approx 624{,}960 \text{ bps} \quad (5.17)$$

Delay and jitter values for each message are given in Appendix I Table A-3. A figure that shows the maximum and mean values of delays and jitters with respect to static slot IDs are given in Figure 6-8.

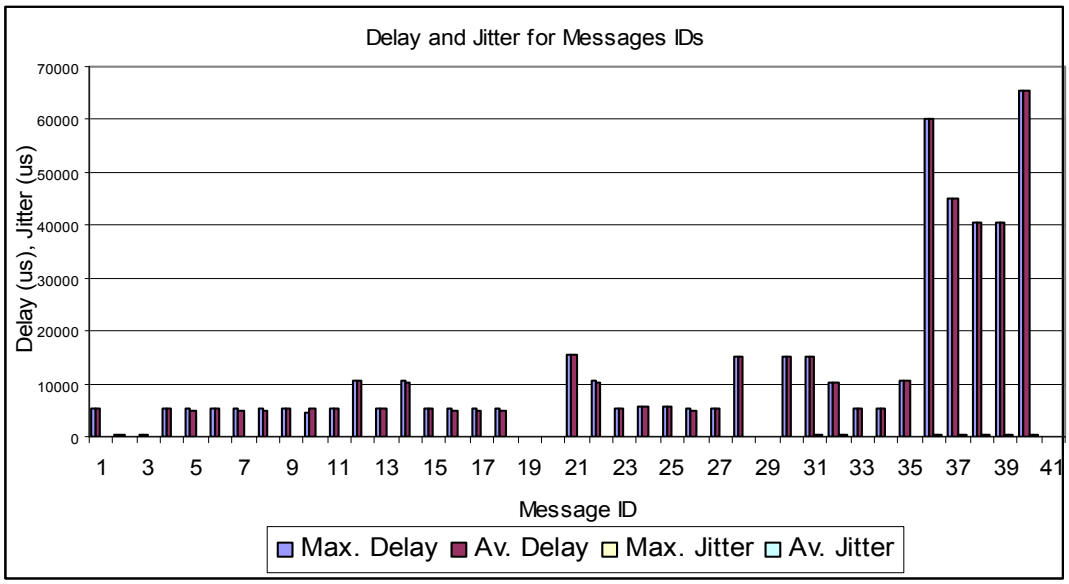

Figure 6-8: Experiment 3 – delay (µsec) & jitter (µsec) vs. Message ID

The behavior of the delay and jitter values for the Message IDs are the same as in Experiment 2. This experiment shows that cycle multiplexing can be used to decrease U_{ID} without affecting the delay and jitter of the messages. By this way more slots are freed for the possible addition of new nodes to the cluster. In other words more nodes can be connected to the FlexRay network without changing configuration of the static segment. This allocation method adds more flexibility to the network.

6.1.4 Experiment 4: 3 Nodes, Cycle Multiplexing with Jitter

In experiments 2 and 3, to avoid the jitter the required repetition time value for the frame factorized and the largest possible factor that is power of two and less than or equal to 64 is chosen as mentioned in Section 5.1.2. In this experiment to further

improve the utilization and U_{ID} jitter is allowed. Static slot allocation of the new configuration is given in Table 6-8. Assignment of the messages to the static slots is performed according to the method defined in [21].

Table 6-8: Experiment 4: Message Specific Configuration Parameters

Message ID	Segment	Static Slot	Repetition	Offset
Message 1	Static	12	2	0
Message 2	Static	11	1	0
Message 3	Static	16	4	1
Message 4	Static	12	2	1
Message 5	Static	7	2	0
Message 6	Static	13	2	0
Message 7	Static	7	2	1
Message 8	Static	8	2	0
Message 9	Static	13	2	1
Message 10	Static	14	2	0
Message 11	Static	14	2	1
Message 12	Static	17	4	0
Message 13	Static	15	2	0
Message 14	Static	16	4	3
Message 15	Static	15	2	1
Message 16	Static	8	2	1
Message 17	Static	9	2	0
Message 18	Static	9	2	1
Message 19	Static	10	16	0
Message 20	Static	10	8	1
Message 21	Static	17	16	1
Message 22	Static	17	16	2
Message 23	Static	17	16	3
Message 24	Static	18	32	0
Message 25	Static	18	64	1
Message 26	Static	10	32	2
Message 27	Static	16	2	0
Message 28	Static	10	16	3
Message 29	Static	10	16	4
Message 30	Static	10	16	5
Message 31	Static	10	64	6
Message 32	Static	10	64	7
Message 33	Static	10	64	8
Message 34	Static	10	64	10
Message 35	Static	18	4	2
Message 36	Static	10	64	11
Message 37	Static	10	64	12
Message 38	Static	17	64	5
Message 39	Static	17	64	6
Message 40	Static	18	64	3
Message 41	Static	10	16	13

In this configuration used number of static slots is reduced to

$$U_{ID} = 12 \tag{5.18}$$

Allocated number of static slots is calculated as

$$SA = 10.34375 \tag{5.19}$$

Then allocated bandwidth is

$$BW_A = (10.34375*31)/5000 = 0.064 = 6.4\% \tag{5.20}$$

Allocated number of static slots per one second is

$$B_A = 200*10.34375 = 2068.75 \tag{5.21}$$

Total number of messages transmitted during the experiment is 125443. This corresponds to $B_U = 2016$ messages per second. Then Utilization is given as

$$U = 2016/2068.75 = 97.5\% \tag{5.22}$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.975 * 0.064 \approx 624{,}960 \text{ bps} \tag{5.23}$$

Delay and jitter values for each message is given in Appendix I Table A-4. A figure that shows the maximum and mean values of delays and jitters with respect to message IDs are given in Figure 6-9.

In this experiment there is a significant increase for the delay and jitter values as expected. By this method network utilization is increased, U_{ID} and SA are decreased in the expense of more delay and jitter. It will be the system designer's choice which scheduling method should be selected (without jitter or with jitter) by considering requirements of the messages.

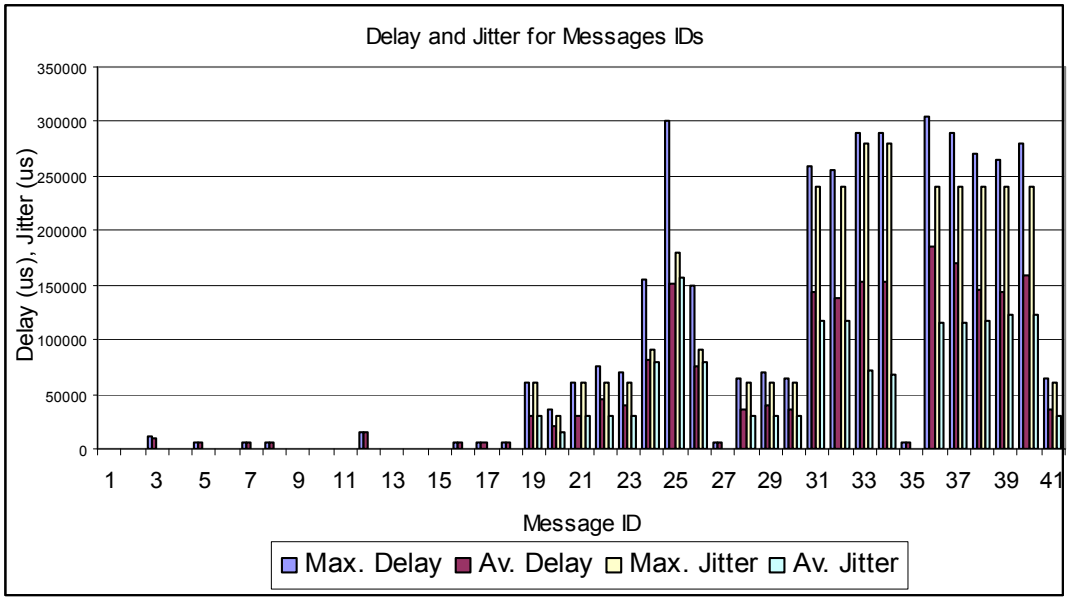

Figure 6-9: Experiment 4 – delay (µsec) & jitter (µsec) vs. Message ID

6.1.5 Experiment 5: 3 Nodes, Cycle Multiplexing without Jitter – Signal Packing

In this experiment signal packing of the payload of messages which have the same periods and node number is performed as expressed in Section 3.4 to further decrease BW_U, U_{ID} and SA without affecting the delay and jitter values. The resulting maximum payload length of the static segment for the given message set (Table 6-4) is calculated to be 24 bytes. Message map according to the 24 bytes payload for the first 3 Nodes is given in Table 6-9. Packing of the messages into frames and assignment of frames to the static slots are performed according to the method defined in [21]. Packed signal data is generated at the same time and transmitted in the same FlexRay frame. Number of FlexRay frames is reduced to 20 by packing. In the previous experiments every message payload (signal) is transmitted in a separate frame.

Table 6-9: Experiment 5: Message Map and Configuration Parameters for the Network with 3 Nodes

Message ID	Message Period	Total Message Size	Sender Node	Slot ID	f_{RT}	Offset	Receiver Node
Message 5_7_8	10 ms	24 bytes	Node 1	7	2	0	Analyzer
Message 16_17_18	10 ms	20 bytes	Node 1	7	2	1	Analyzer
Message 20	50 ms	8 bytes	Node 1	8	2	0	Analyzer
Message 19_28_29	100 ms	16 bytes	Node 1	9	4	0	Analyzer
Message 30_41	100 ms	12 bytes	Node 1	9	4	1	Analyzer
Message 26	250 ms	3 bytes	Node 1	8	2	1	Analyzer
Message 33_34	1000 ms	8 bytes	Node 1	9	8	2	Analyzer
Message 31_32_36	2000 ms	24 bytes	Node 1	9	16	3	Analyzer
Message 37	2000 ms	8 bytes	Node 1	9	16	6	Analyzer
Message 2	5 ms	8 bytes	Node 2	10	1	0	Analyzer
Message 1_4_6	10 ms	24 bytes	Node 2	11	2	0	Analyzer
Message 9_10_11	10 ms	24 bytes	Node 2	11	2	1	Analyzer
Message 13_15_27	10 ms	24 bytes	Node 2	12	2	0	Analyzer
Message 3_12_14	20 ms	24 bytes	Node 2	12	4	1	Analyzer
Message 21_22_23	100 ms	18 bytes	Node 2	12	4	3	Analyzer
Message 38_39	2000 ms	16 bytes	Node 2	13	16	0	Analyzer
Message 24	250 ms	2 bytes	Node 3	15	4	0	Analyzer
Message 25	500 ms	1 bytes	Node 3	14	2	0	Analyzer
Message 35	20 ms	8 bytes	Node 3	15	4	1	Analyzer
Message 40	2000 ms	8 bytes	Node 3	15	16	2	Analyzer

Used number of Static Slots and Static Slot Allocation for the packed message set is calculated as:

$$U_{ID} = 9 \tag{5.24}$$

$$SA = 6.875 \tag{5.25}$$

Total allocated bandwidth per cycle for 3 Nodes is

$$BW_A = (6.875 * 45)/5000 \approx 0.062 \quad (5.26)$$

Allocated number of static slot per one second is

$$B_A = 200*6.875 = 1375 \quad (5.27)$$

Transmitted frames with packed messages per second is B_U = 863 frames per second. Then Utilization is calculated as

$$U = 863/1375 = 63\% \quad (5.28)$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.062 * 0.63 = 388{,}350 \text{ bps} \quad (5.29)$$

Maximum and average delay values for each Message ID are given in Figure 6-10 and detailed results are given in Table A-38.

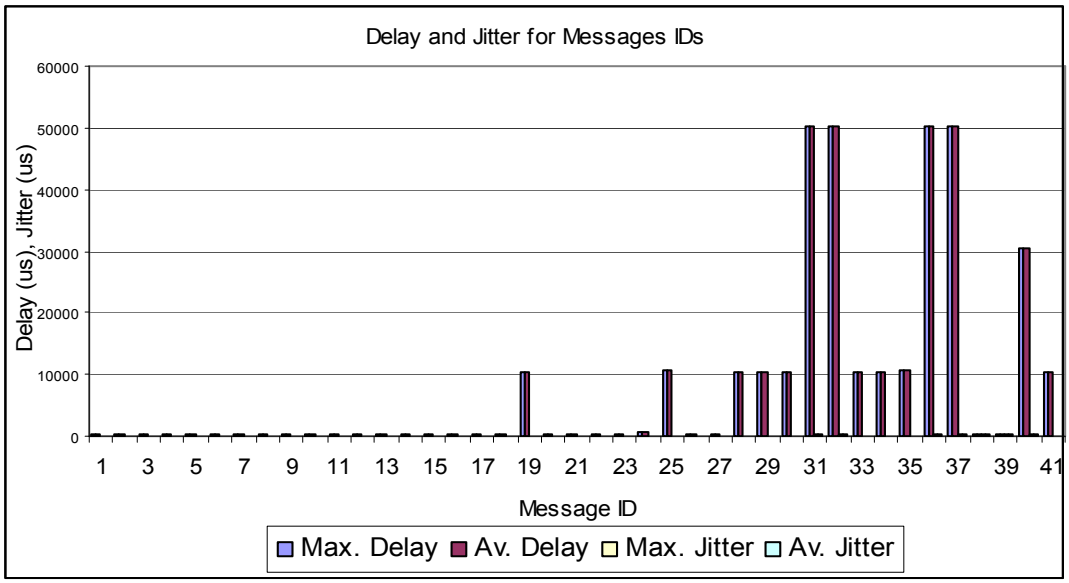

Figure 6-10: Experiment 5 – delay (µsec) & jitter (µsec) vs. Message ID

As a result there is a significant decrease in U_{ID}, SA, and BW_U values as compared to experiment 3. Since duration of a static slot is increased, measured utilization decreases. However the same message set is transmitted by using less bandwidth.

Therefore more messages can be transmitted in a single network by applying packing as compared to the previous experiments. Furthermore maximum delay values that are observed during the experiment are reduced. The reason for this decrease is the synchronous generation of the messages with scheduling offsets.

6.1.6 Experiment 6: 6 Nodes, Cycle Multiplexing without Jitter

In section 6.1.3, used number of static slot IDs is optimized by using cycle multiplexing. The message schedule is computed by taking into account the message jitters and jitter is kept at minimum as in Experiment 3. The same methodology is also followed in this experiment with additional 3 nodes and 10 messages. Scheduling parameters for the additional messages are given in Table 6-10. Assignment of the messages to the static slots is performed according to the method defined in [21]. Values of the parameters are the same as in Table 6-7 for the first 41 messages.

Table 6-10: Experiment 6: Message specific parameters for new messages

Message ID	Segment	Static Slot	Repetition	Offset
Message 42	Static	23	4	0
Message 43	Static	23	4	1
Message 44	Static	23	4	2
Message 45	Static	24	2	0
Message 46	Static	24	2	1
Message 47	Static	25	4	0
Message 48	Static	26	2	0
Message 49	Static	26	2	1
Message 50	Static	27	4	0
Message 51	Static	27	4	1

Used number of static slot IDs is increased by 5 for the new 10 messages is

$$U_{ID} = 16 + 5 = 21 \tag{5.30}$$

Additional allocated number of static slots equals to 3.5, then

$$SA = 13.9375 + 3.5 = 17.4375 \quad (5.31)$$

Total allocated bandwidth for 6 nodes is

$$BW_A = (17.4375*31)/5000 = 0.108 = 10.8\% \quad (5.32)$$

Allocated number of static slot per one second is

$$B_A = 200*17.4375 = 3487.5 \quad (5.33)$$

Total number of messages transmitted during the experiment is 143404. This corresponds to $B_U = 2292$ messages per second. Then Utilization is calculated as

$$U = 2292/3487.5 = 65.7\% \quad (5.34)$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.108 * 0.657 = 710{,}520 \text{ bps} \quad (5.35)$$

Delay and jitter values for each message are given in Appendix I Table A-10. A figure that shows the maximum and mean values of delays and jitters with respect to message IDs are given in Figure 6-11.

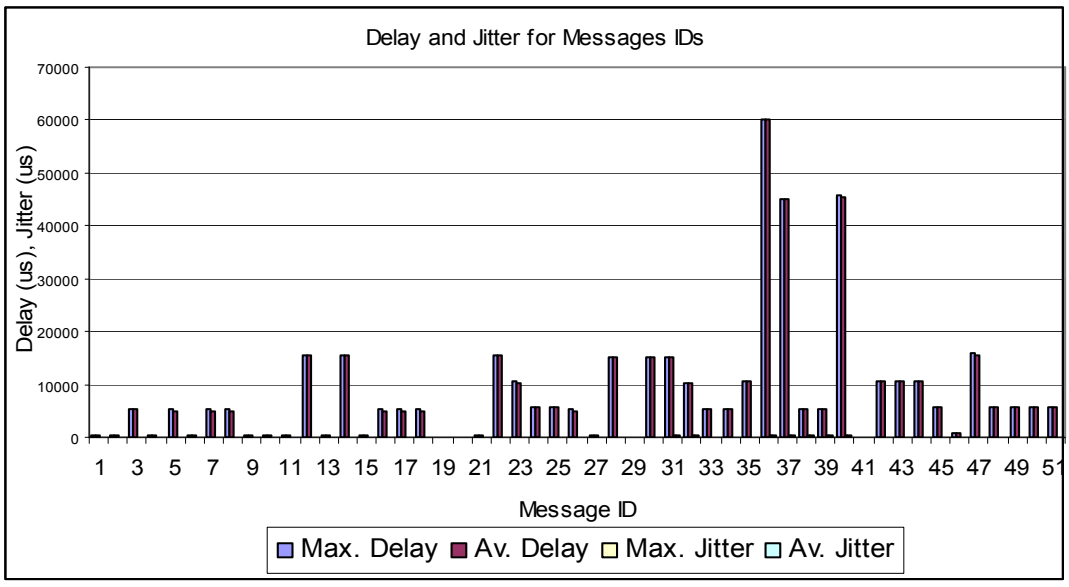

Figure 6-11: Experiment 6 – delay (µsec) & jitter (µsec) vs. Message ID

The delay and jitter values in this experiment show the same behavior as in the Experiment 3. The only difference is that the number of nodes and messages are increased.

6.1.7 Experiment 7: 6 Nodes, Cycle Multiplexing with Jitter

Jitter is allowed to decrease used static slot IDs in this experiment as in Experiment 4. Message specific configuration parameters for the messages 1 to 41 are the same as in Table 6-8. For the messages 42 to 51 these parameters are given in Table 6-11. Assignment of the messages to the static slots is performed according to the method defined in [21].

Table 6-11: Experiment 7: Message Specific parameters for new messages

Message ID	Segment	Static Slot	Repetition	Offset
Message 42	Static	19	4	1
Message 43	Static	19	64	3
Message 44	Static	19	16	7
Message 45	Static	19	32	11
Message 46	Static	19	2	0
Message 47	Static	20	64	1
Message 48	Static	20	32	3
Message 49	Static	20	2	0
Message 50	Static	21	64	0
Message 51	Static	21	64	1

Used number of static slot IDs is increased by 3 for the additional 10 messages, then

$$U_{ID} = 12+ 3 = 15 \tag{5.36}$$

Then additional allocated number of static slots equals to 1.4375, then

$$SA = 10.34375 + 1.4375 = 11.78125 \tag{5.37}$$

Total allocated bandwidth for 6 nodes is

$$BW_A = (11.78125*31)/5000 = 0.073 = 7.3\% \quad (5.38)$$

Allocated number of static slots in one second is

$$B_A = 200*11.78125 = 2356.25 \quad (5.39)$$

Total number of messages transmitted during the experiment is 149781. This corresponds to $B_U = 2292$ messages per second. Then Utilization is calculated as

$$U = 2292/2356.25 = 97.3\% \quad (5.40)$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.073 * 0.973 \approx 710{,}520 \text{ bps} \quad (5.41)$$

The results of delay and jitter analysis are given in Appendix I Table A-11. A figure that shows the maximum and mean values of delays and jitters with respect to Message IDs are given in Figure 6-12.

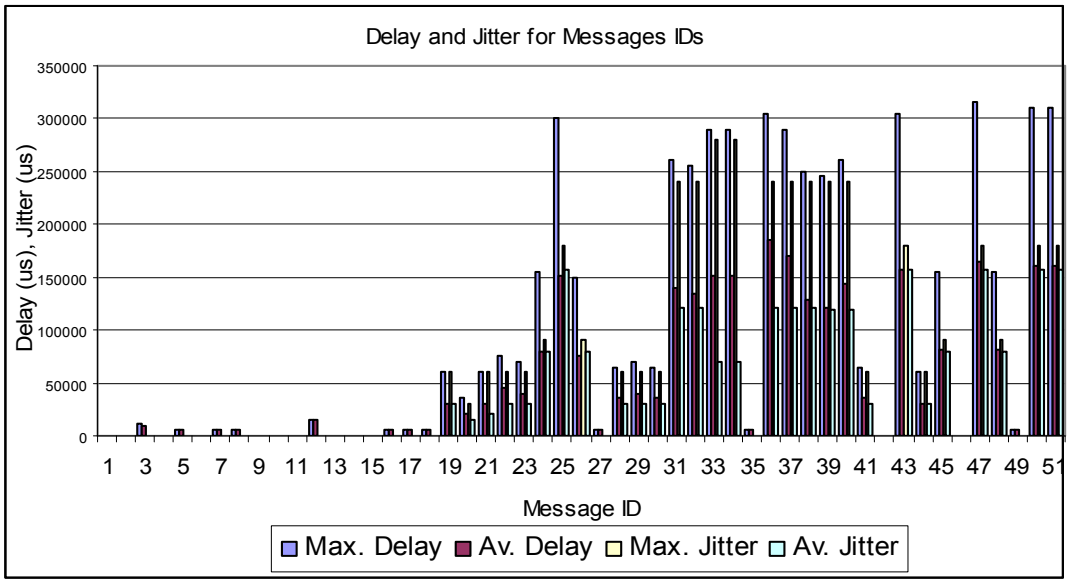

Figure 6-12: Experiment 7 – delay (µsec) & jitter (µsec) vs. Message ID

Results show that the same behavior of the delay and jitter values of Experiment 4 is followed because of the deterministic structure of the static segment.

6.1.8 Experiment 8 and 9: 6 Nodes, Cycle Multiplexing without Jitter – 2.5ms FlexRay Cycle

In experiments 8 and 9, FlexRay cycle duration is reduced to 2.5 ms. Message generation by the microcontroller of the evaluation board is performed in two ways for each experiment. While generation times are synchronized with scheduling offset values for each message in Experiment 8, first occurrence of messages that belong to a given node is set to be the same cycle (random) in Experiment 9. In other words messages are transmitted at the cycles that the message is generated in Experiment 8, and delay values are reduced by this method. In experiment 9 the delay values are expected to be larger. All the other variables including message specific configuration parameters (Table 6-12) and cluster configuration parameters (Table 6-3) are kept constant. For this reason performance metrics other than delay values are the same for both experiments. Assignment of the messages to the static slots is performed according to the method defined in [21].

Table 6-12: Experiment 8 and 9: Message Specific Configuration Parameters

Message ID	Segment	Static Slot	Repetition (f_{RT})	Scheduling Offset
Message 1	Static	7	4	0
Message 2	Static	7	2	1
Message 3	Static	10	8	2
Message 4	Static	7	4	2
Message 5	Static	8	4	0
Message 6	Static	9	4	0
Message 7	Static	8	4	1
Message 8	Static	8	4	2
Message 9	Static	9	4	1
Message 10	Static	9	4	2
Message 11	Static	9	4	3
Message 12	Static	10	8	3
Message 13	Static	10	4	0
Message 14	Static	10	8	6
Message 15	Static	10	4	1
Message 16	Static	8	4	3
Message 17	Static	11	4	0
Message 18	Static	11	4	1
Message 19	Static	13	8	0

Table 6-12 (continued)

Message 20	Static	11	4	2
Message 21	Static	10	8	7
Message 22	Static	12	8	1
Message 23	Static	12	8	2
Message 24	Static	14	4	0
Message 25	Static	14	8	1
Message 26	Static	11	4	3
Message 27	Static	12	4	0
Message 28	Static	13	8	1
Message 29	Static	13	8	2
Message 30	Static	13	8	3
Message 31	Static	13	32	5
Message 32	Static	13	32	6
Message 33	Static	13	16	7
Message 34	Static	13	16	13
Message 35	Static	14	8	2
Message 36	Static	13	32	14
Message 37	Static	13	32	15
Message 38	Static	12	32	3
Message 39	Static	12	32	5
Message 40	Static	14	32	3
Message 41	Static	13	8	4
Message 42	Static	15	8	2
Message 43	Static	15	8	3
Message 44	Static	15	8	6
Message 45	Static	15	4	0
Message 46	Static	15	4	1
Message 47	Static	16	8	2
Message 48	Static	16	4	0
Message 49	Static	16	4	1
Message 50	Static	17	8	0
Message 51	Static	17	8	1

Used number of static slot IDs and Static Slot Allocation is calculated as

$$U_{ID} = 11 \tag{5.42}$$

$$SA = 8.71875 \tag{5.43}$$

Total allocated bandwidth for 6 nodes is

$$BW_A = (8.71875*31)/2500 = 0.108 = 10.8\% \tag{5.44}$$

Allocated number of static slots in one second is

$$B_A = 400*8.71875 = 3487.5 \quad (5.45)$$

Total number of messages transmitted during the experiment is 149781. This corresponds to $B_U = 2292$ messages per second. Then Utilization is calculated as

$$U = 2292/3487.5 = 65.7\% \quad (5.46)$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.108 * 0.657 \approx 710{,}520 \text{ bps} \quad (5.47)$$

A figure that shows the maximum and average values of delays and jitters with respect to message IDs for both experiments are given in Figure 6-13 and Figure 6-14, and detailed results are given in Table A-39 and Table A-40.

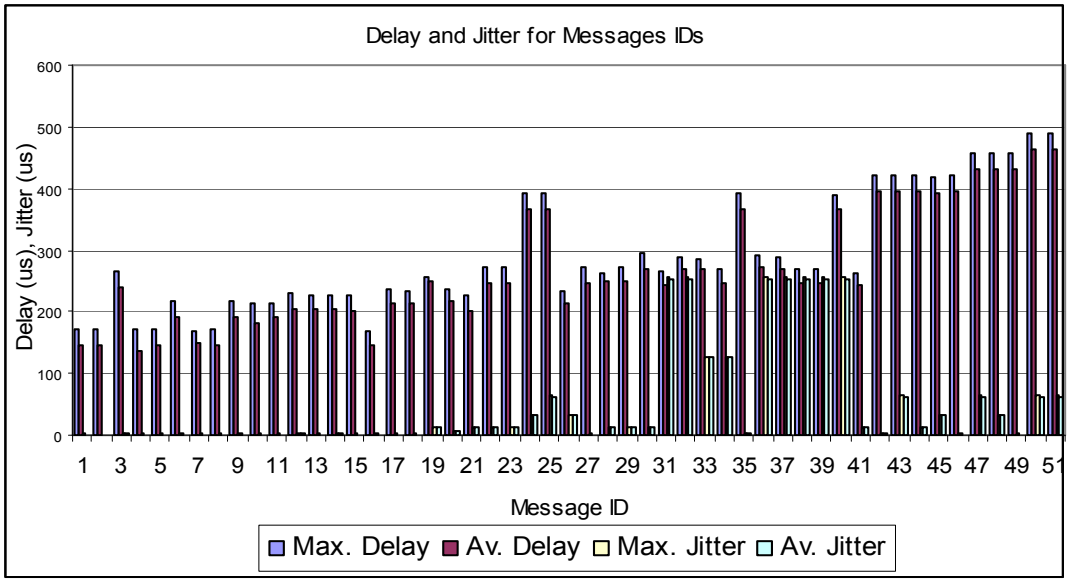

Figure 6-13: Experiment 8 – delay (µsec) & jitter (µsec) vs. Message ID (message generations are synchronized with scheduling offsets)

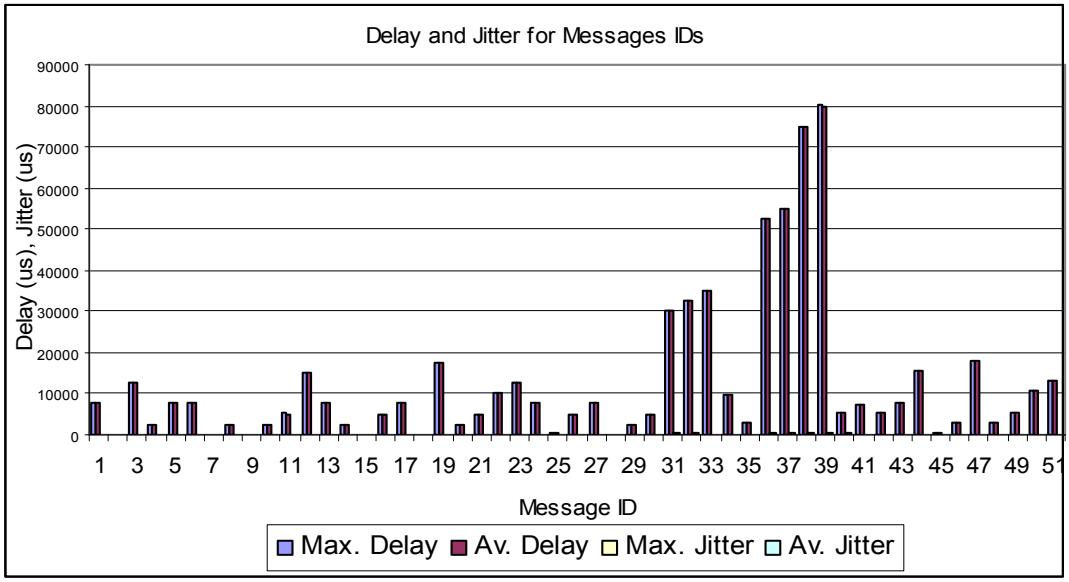

Figure 6-14: Experiment 9 – delay (µsec) & jitter (µsec) vs. Message ID (first occurrence of the messages that belongs to a specific node happens in the same cycle)

U_{ID} and SA values are half of the values in Experiment 6. Since the cycle duration and available bandwidth in one cycle is also half of the Experiment 6 values, utilization is calculated to be the same. By reducing the cycle duration only delay values can be improved (decreased). If the same number of static slots in a single cycle is used as compared to 5 ms cycle duration, dynamic segment duration would be reduced. In our experiment number of static slots is also reduced to half to preserve the ratio of dynamic segment duration to cycle duration.

6.1.9 Experiment 10: 6 Nodes, Cycle Multiplexing without Jitter – Signal Packing

In this experiment by considering the 6 Nodes, packing of the messages which have the same periods and node number is performed according to [21] to further increase the utilization and decrease U_{ID}, SA and BW_U. The maximum payload length of the static segment for the given message set (Table 6-4) is calculated to be 8 bytes. Message map according to the 8 bytes payload for 6 Nodes is shown in Table 6-13. Packing of the messages into frames and assignment of frames to the static slots are performed according to the method defined in [21]. Messages with

IDs 29-30, 33-34, and 50-51 is packed into the same frame. Packed messages are generated at the same time and transmitted in the same frame. Number of discrete frames is reduced from 51 to 48 by packing method.

Table 6-13: Experiment 10: Message Map and Configuration Parameters for the Network with 3 Nodes

Message ID	Message Period	Total Message Size	Sender Node	Slot ID	f_{RT}	Offset	Receiver Node
Message 5	10 ms	8 bytes	Node 1	7	2	0	Analyzer
Message 7	10 ms	8 bytes	Node 1	7	2	1	Analyzer
Message 8	10 ms	8 bytes	Node 1	8	2	0	Analyzer
Message 16	10 ms	8 bytes	Node 1	8	2	1	Analyzer
Message 17	10 ms	8 bytes	Node 1	9	2	0	Analyzer
Message 18	10 ms	4 bytes	Node 1	9	2	1	Analyzer
Message 20	50 ms	8 bytes	Node 1	10	2	0	Analyzer
Message 19	100 ms	8 bytes	Node 1	11	4	0	Analyzer
Message 28	100 ms	4 bytes	Node 1	11	4	1	Analyzer
Message 29_30	100 ms	8 bytes	Node 1	11	4	2	Analyzer
Message 41	100 ms	8 bytes	Node 1	11	4	3	Analyzer
Message 26	250 ms	3 bytes	Node 1	10	2	1	Analyzer
Message 33_34	1000 ms	8 bytes	Node 1	12	8	0	Analyzer
Message 31	2000 ms	8 bytes	Node 1	12	16	1	Analyzer
Message 32	2000 ms	8 bytes	Node 1	12	16	2	Analyzer
Message 36	2000 ms	8 bytes	Node 1	12	16	3	Analyzer
Message 37	2000 ms	8 bytes	Node 1	12	16	4	Analyzer
Message 2	5 ms	8 bytes	Node 2	13	1	0	Analyzer
Message 1	10 ms	8 bytes	Node 2	14	2	0	Analyzer
Message 4	10 ms	8 bytes	Node 2	14	2	1	Analyzer
Message 6	10 ms	8 bytes	Node 2	15	2	0	Analyzer
Message 9	10 ms	8 bytes	Node 2	15	2	1	Analyzer
Message 10	10 ms	8 bytes	Node 2	16	2	0	Analyzer
Message 11	10 ms	8 bytes	Node 2	16	2	1	Analyzer
Message 13	10 ms	8 bytes	Node 2	17	2	0	Analyzer
Message 15	10 ms	8 bytes	Node 2	17	2	1	Analyzer
Message 27	10 ms	8 bytes	Node 2	18	2	0	Analyzer
Message 3	20 ms	8 bytes	Node 2	18	4	1	Analyzer
Message 12	20 ms	8 bytes	Node 2	18	4	3	Analyzer
Message 14	20 ms	8 bytes	Node 2	19	4	0	Analyzer
Message 21	100 ms	2 bytes	Node 2	19	4	1	Analyzer
Message 22	100 ms	8 bytes	Node 2	19	4	2	Analyzer
Message 23	100 ms	8 bytes	Node 2	19	4	3	Analyzer
Message 38	2000 ms	8 bytes	Node 2	20	16	0	Analyzer
Message 39	2000 ms	8 bytes	Node 2	20	16	1	Analyzer
Message 35	20 ms	8 bytes	Node 3	22	4	0	Analyzer
Message 24	250 ms	2 bytes	Node 3	21	2	0	Analyzer
Message 25	500 ms	1 bytes	Node 3	22	4	1	Analyzer
Message 40	2000 ms	8 bytes	Node 3	22	16	2	Analyzer

Table 6-13 (continued)

Message 46	10 ms	4 bytes	Node 4	23	2	0	Analyzer
Message 42	20 ms	2 bytes	Node 4	24	4	0	Analyzer
Message 44	100 ms	2 bytes	Node 4	24	4	1	Analyzer
Message 45	250 ms	8 bytes	Node 4	23	2	1	Analyzer
Message 43	500 ms	1 byte	Node 4	24	4	2	Analyzer
Message 49	10 ms	4 bytes	Node 5	25	2	0	Analyzer
Message 48	250 ms	8 bytes	Node 5	25	2	1	Analyzer
Message 47	500 ms	8 bytes	Node 5	26	4	0	Analyzer
Message 50_51	500 ms	8 bytes	Node 6	27	4	0	Analyzer

Used number of Static Slots and Static Slot Allocation for the packed message set is calculated as:

$$U_{ID} = 21 \tag{5.48}$$

$$SA = 16.8125 \tag{5.49}$$

Total allocated bandwidth per cycle for 6 Nodes is

$$BW_A = (16.8125 * 29)/5000 \approx 0.098 \tag{5.50}$$

Allocated number of static slot per one second is

$$B_A = 200*16.8125 = 3362.5 \tag{5.51}$$

Transmitted messages per second is B_U = 2279 messages per second. Then Utilization is calculated as

$$U = 2279/3362.5 = 68\% \tag{5.52}$$

Used bandwidth is

$$BW_U = 10 \text{ Mbps} * 0.062 * 0.63 = 660{,}910 \text{ bps} \tag{5.53}$$

Maximum and average delay values for each Message ID are given in Figure 6-15 and detailed results are given in Table A-41.

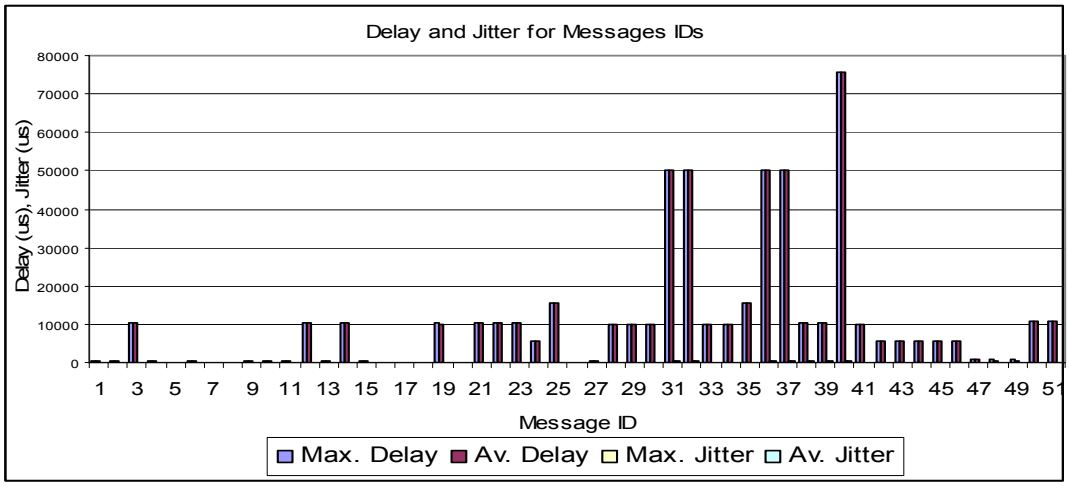

Figure 6-15: Experiment 10 – delay (µsec) & jitter (µsec) vs. Message ID

Delay and jitter values are following the same trend as compared to Experiment 6 values. Message packing improved the performance metrics related with network usage which are SA, BW_U and utilization without any degradation for delay and jitter values.

6.1.10 Summary and Discussion of the Results of the Static Segment Experiments

In the static segment experiments, we investigated the effect of the scheduling of the messages and the number of nodes in the network on the performance metrics that are introduced in Section 3.2. One slot is allocated for each message in experiments 1 and 2. However in Experiment 2, cycle filtering is used to reduce the U_{ID} and SA, and also to increase the utilization. By this way slots are made partially empty for the possible addition of new messages for the same node. Addition of new messages can be satisfied by updating the configuration of the related node only without affecting the whole network. As compared to Experiment 1 utilization increase to 72%. In Experiment 1 utilization is only 24.6%. In both experiments jitter for individual messages is the same. However delay values increased for the reason given in Section 6.1.2. Although delay values increased, there is still a limit value for the maximum delay because of the scheduling method. This limit is

"Max. Delay < (f_{RT} – *1)*gdCycle + SlotID*gdStaticSlot* gdMacrotick"* and it is valid for all of the static segment experiments. By this way on time message transmissions can be guaranteed.

When a new node is to be added to an existing network, a different slot has to be allocated for it. The worst case for static slot allocation is realized in the Experiment 1. Then some cycles are freed for other message transmissions of the same node in the Experiment 2 by using cycle filtering. After that, in the Experiment 3 empty cycles are filled with present messages and more static slots are made available for new messages and also for new nodes. That means U_{ID} is reduced significantly. Jitter is the same in experiments 1 and 2 as the slots that the messages are transmitted are the same, and the above given *Max. Delay* definition is applicable for this experiment.

In experiment 4, jitter is allowed and static slot allocation and repetition times changed accordingly. It is calculated that there is a significant increase for the delay and jitter values in return for increased utilization and reduced U_{ID} and *SA*. In experiment 5 U_{ID}, *SA*, BW_U is decreased without affecting the delay and jitter values. That means drawback of the method given in experiment 4 can be overcome by applying packing.

The delay and jitter values experimentally measured by considering all static segment messages in the experiments are given in the Table 6-14. This table shows the outcomes of each scheduling method clearly.

Table 6-14: Performance metrics calculated from the log data – Delay and Jitter

E	N	U_{ID}	Av. *J* (µs)	Av. *J* %	Max. *J* (µs)	Max. *J* %	Av. *D* (µs)	Av. *D* %	Max. *D* (µs)	Max. *D* %
1	**3**	41	53.81	0.01	243	0.02	662	3.07	1370	26.76
2	**3**	41	54.74	0.01	248	0.02	11882	15.4	71027	63.08
3	**3**	16	53.66	0.01	242	0.02	11615	25.22	65628	54.01

Table 6-14 (continued)

4	**3**	12	37553	10.25	279839	59.98	50811	25.68	305079	77.01
5	**3**	9	55.83	0.01	256	0.02	8144	10.5	50295	44.6
6	**6**	21	50.16	0.01	248	0.02	8390.55	15.47	60145	53.41
7	**6**	15	46377.5	12.36	279842	59.98	56015.78	26.18	315569	77.01
8	**6**	11	51.2	0.01	254	0.02	266.7	0.4	491	0.6
9	**6**	11	51.2	0.01	254	0.02	12356	18.5	80029	71.0
10	**6**	21	50.6	0.01	251	0.02	10227	15.3	75576	67.1

In the table, E is Experiment Number, N is Number of Nodes, U_{ID} is used number of static slot IDs, J is jitter, and D is delay. The percentages shown are calculated according to the deadline values of each message which is the period of that message. In Table 6-15 and Table 6-16 U_{ID}, SA, and BW_A are calculated from the scheduling tables, *Utilization* and BW_U are measured from the experiment outputs. The comparison of the Performance Metrics for the first 3 nodes experiments is given in Table 6-15.

Table 6-15: Comparison of the Performance Metrics for the 3 Nodes Experiments – Static Segment

Experiment	Used Static Slot ID (U_{ID})	Static Slot Allocation (SA)	Allocated Bandwidth (BW_A)	Utilization (U)	Used Bandwidth (BW_U)
Experiment 1	41	41	25.42 %	24.6 %	624,960 bps
Experiment 2	41	13.9375	8.64 %	72 %	624,960 bps
Experiment 3	16	13.9375	8.64 %	72 %	624,960 bps
Experiment 4	12	10.34375	6.4 %	97.5 %	624,960 bps
Experiment 5	9	6.875	6.2 %	63 %	388,350 bps
*)	8	6,96875	8.64 %	72 %	624,960 bps

***) Shows the results of 2.5 ms cycle duration experiments (Experiments 8 and 9) by considering first 3 nodes**

The methods used in experiments 6 and 7 are the repetition of the methods used in experiments 3 and 4 respectively, and also the results follow the same rule. These experiments are performed to show the applicability of the methods in a relatively larger network configuration.

In the experiments 8 and 9 cycle duration is reduced to 2.5 ms. By this way the time between two consecutive instances of a given slot ID is reduced to 2.5 ms, therefore delay values is expected to be lower. In the experiment 8 messages are set to be generated at the cycles in which they are allowed to be transmitted. This case gives the minimum delay values for each message. In the experiment 9, first occurrence of the messages that belongs to the same period is set to be the same cycle. This scheme causes an increase in delay values. Since the repetition times are doubled when the cycle duration is reduced to the half, all the performance metrics related with the network usage and allocation (U_{ID}, *SA, Utilization*, BW_U) have the same values of Experiment 6.

In the experiment 10 message packing is applied as in experiment 5 to decrease the U_{ID}, *SA*, and BW_U and increase the utilization. The results conform to the expectations.

In the experiments 1, 2, 3, 5, 6, 8, 9 and 10 the jitter values are expected to be zero because periods of the messages are integer multiples of cycle duration and static slots is allocated also periodically. However jitter is more than zero, because of the clock drifts resulted in the local oscillators of evaluation boards. This drift is approximately 3 μs in every 2 FlexRay cycles. Message IDs with higher periods are experiencing more jitter, because there are more cycles between consecutive messages.

The comparison of the performance metrics for the 6 Nodes experiments are given in the Table 6-16.

Table 6-16: Comparison of the Performance Metrics for the 6 Nodes Experiments – Static Segment

Experiment	Used Static Slot ID (U_{ID})	Static Slot Allocation (SA)	Allocated Bandwidth (BW_A)	Utilization (U)	Used Bandwidth (BW_U)
*)	51	51	31.62 %	28 %	710,520 bps
**)	51	17.4375	10.8 %	65.7 %	710,520 bps
Experiment 6	21	17.4375	10.8 %	65.7 %	710,520 bps
Experiment 7	15	11.78125	7.3 %	97.3 %	710,520 bps
Experiment 8, 9	11	8.71875	10.8 %	65.7 %	710,520 bps
Experiment 10	21	16.8125	9.8 %	68 %	660,910 bps

***) Shows the theoretical results for the case of simplest scheduling method is applied**
****) Shows the theoretical results for the case of cycle filtering without jitter method is applied**

We conclude that, there is a trade off between *utilization*, U_{ID}, *SA;* and jitter and delay values. If we want to improve the former group performance metrics then we need to allow performance degradation for the jitter and delay values or the opposite. These methods can also be applied partially to the messages. Some of the messages can be scheduled according to the method given in experiment 1 if the timing requirements of the message are very strict, and some of them can be scheduled according to the method given in experiment 4 if the timing requirements allow high delay and jitter. It entirely depends on the requirements for the electronic applications. The summary for the properties of each scheduling method applied in the respective experiments and outcome of these methods are given in Table 6-17.

Table 6-17: Summary of the Static Segment Experiments

E	Scheduling Properties	Results
1	3 Nodes, Simplest scheduling method. One slot allocated for each messages in all cycles that means repetition times of the FlexRay frames are all one. (U_{ID} = SA = number of message IDs)	High U_{ID} and SA values and low utilization (inefficient bandwidth usage). Minimum delay and jitter.
2	3 Nodes, Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle filtering is applied to minimize the Static Slot Allocation (SA). Each message assigned to a different static slot ID (U_{ID} = number of message IDs).	Low SA and high utilization but high U_{ID}. Minimum jitter but increased delay
3	3 Nodes, Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the SA and U_{ID}.	Low U_{ID} and SA, and high utilization. Minimum jitter but increased delay.
4	3 Nodes, Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and minimum allocation can be achieved. Cycle multiplexing is applied to minimize the SA and U_{ID}.	The best U_{ID} and SA (the lowest), and high utilization. The worst delay and jitter (the highest).
5	3 Nodes, Signal Packing is applied to optimize the bandwidth usage. Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the SA and U_{ID}.	Increased bandwidth efficiency (lower SA), but utilization decreases because of increasing static slot duration as compared to Experiment 3. The same message set is transmitted by using lower bandwidth. Similar delay and jitter as compared to Experiment 3.

Table 6-17 (continued)

6	6 Nodes, Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the *SA* and U_{ID}.	Low U_{ID} and *SA*, and high utilization. Minimum jitter but increased delay.
7	6 Nodes, Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and minimum allocation can be achieved. Cycle multiplexing is applied to minimize the *SA* and U_{ID}.	The best U_{ID} and *SA* (the lowest), and high utilization. The worst delay and jitter (the highest).
8	6 Nodes, Cycle duration reduced to its half to decrease delay values for each message ID. Generation times of the messages are synchronized with allocated transmission cycles to minimize delays. Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the *SA* and U_{ID}.	Lower cycle duration Low U_{ID} and *SA*, and high utilization. Minimum jitter. Delay is minimized by synchronization.
9	6 Nodes, Cycle duration reduced to its half to decrease delay values for each message ID. Generation times of the first instances of each message IDs are at the same cycle. Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the *SA* and U_{ID}.	Lower cycle duration Low U_{ID} and *SA*, and high utilization. Minimum jitter. Delay increases as compared to Experiment 8.
10	6 Nodes, Signal Packing is applied to optimize the bandwidth usage. Repetition times of the FlexRay frames are adjusted so that bandwidth efficiency can be maximized and messages experience minimum jitter. Cycle multiplexing is applied to minimize the *SA* and U_{ID}.	Increased bandwidth efficiency (lower *SA*, higher utilization) as compared to Experiment 6. The same message set is transmitted by using lower bandwidth. Similar delay and jitter as compared to Experiment 6.

6.2 Dynamic Segment Experiments

Dynamic segment experiments are designed to see the effect of cluster and configuration parameters on the delay of each individual message transmitted in the dynamic segment. These cluster parameters are the number of messages, message traffic rate (number of total messages transmitted in one second), massage payload length and configuration parameters are the dynamic segment length and the priorities of the messages. Dynamic segment experiments are also performed first on the 3 node network, because of the reason given in the previous section, and then repeated on the 6 node network with some additional messages.

Sporadic messages are generated by the MCU's timers and each message is independent from each other. These sporadic messages generated randomly by using the srand() and rand() functions of the C programming language. srand() is used to generate different seeds for every different run of experiments, and rand() is used to randomize interarrival times between minimum and maximum interarrival times of the messages. 6 of 16 Bits Reload timers and 8 of 16 Bits Free Run timers of the MCU is used for the message generation. Counter clock for the Reload Timer has a frequency of 125 kHz and counter clock for the Free Run Timer has a frequency of 250 kHz. Therefore the maximum interrupt times for the timers and also maximum interarrival times for the sporadic messages are 524.28 ms (65535*(1/125 kHz)) for Reload Timers and 262.14 ms (65535*(1/250 kHz)) for the Free Run Timers. The minimum interarrival time for all of the messages is set to be 50 ms which is also defined to be deadline value for the messages. Maximum interarrival times are adjusted to manipulate the message traffic rate in the related experiments (shorter maximum interarrival times means high traffic rate).

Dynamic segment messages are assigned to dynamic slots starting from the first dynamic slot of the dynamic segment. Numbering of the dynamic slots continue from the last Static Slot number, therefore for the case of 64 static slots the number of the first dynamic slot is 65. For a given dynamic slot, the dynamic slot number is also used as the identification for the sporadic message which is allocated to that

dynamic slot. The smallest message ID or Dynamic Slot has the highest priority. All the results for the delay calculations and also comparisons are presented with respect to the message IDs.

Experiments are performed under two different FlexRay cluster configurations. In the first part the limits of the protocol in terms of delay values are explored in both 3 node network and 6 node network. In this respect, payload length is increased gradually then dynamic segment length is reduced and finally message traffic load is increased step by step until delays of the messages are significantly increased. The values of the configuration variables for the first part are given in the Table 6-18. Minislot duration is set to be 6 MT without any concern for the performance metrics. Number of minislots value is the maximum value which corresponds to the remaining time duration after fixing the Static Segment and Network Idle Time. These maximum values are 472 for 64 static slots, 307 for 96 static slots, and 141 for 128 static slots by considering that gdStaticSlot is 31 MT. Unless it is mentioned in the related experiments these values are fixed through all the first part experiments.

Table 6-18: Configuration parameters for dynamic segment – first part of the experiments

Configuration Parameter Name	Value
gdCycle (µs)	5000 µs
gNumberOfMiniSlots	472
gdMinislot (MT)	6 MT
gdMiniSlotActionPointOffset (MT)	2 MT
gdDynamicSlotIdlePhase (Minislot)	1 Minislots
Dynamic Segment Duration	2838 MT
gdNIT (MT)	178 MT
gdMacrotick (µs)	1 µs

In the second part, the effect of payload length, number of messages, dynamic segment duration, and message traffic rate is examined for each priority level in the 6 node network configuration. The configuration variables for this part are given in Table 6-19.

Table 6-19: Configuration parameters for dynamic segment – second part of the experiments

Configuration Parameter Name	Value
gdCycle (µs)	5000 µs
gNumberOfMiniSlots	307
gdMinislot (MT)	6 MT
gdMiniSlotActionPointOffset (MT)	2 MT
gdDynamicSlotIdlePhase (Minislot)	1 Minislots
Dynamic Segment Duration	1848 MT
gdNIT (MT)	178 MT
gdMacrotick (µs)	1 µs

6.2.1 Part 1 – Exploring the Values for Message Delays

The parameters for the experiments with 3 nodes and 6 nodes network configurations are given in the Table 6-20. Observed maximum values for the delays during the experiments are recorded as the maximum delay for each message ID, and average delay value for each message ID is calculated as the sum of delay values of all transmitted messages divided by number of transmitted messages.

In the first experiment all 15 messages have 8 bytes of payload data. As FlexRay has a high bandwidth, we choose to use messages with large data length to make use of this bandwidth. For this reason as a starting point 8 bytes of payload data is chosen for the messages. Then starting from the first dynamic slot messages are assigned. The message IDs are taken as the number of that dynamic slot.

Table 6-20: Designed Experiments and Cluster Configuration Parameters – Part 1

Experiment Number	Number of Nodes	Number of Messages	Dynamic segment length	Payload Length	Message Traffic Rate (Messages/Sec)
1	3	15	472 Minislots	8 Bytes	53
2	3	15	472 Minislots	64 Bytes	53
3	3	15	472 Minislots	254 Bytes	53
4	3	15	141 Minislots	254 Bytes	53
5	3	15	141 Minislots	254 Bytes	300
6	6	36	472 Minislots	16 Bytes	188
7	6	36	472 Minislots	64 Bytes	188
8	6	36	472 Minislots	254 Bytes	188
9	6	36	141 Minislots	254 Bytes	188
10	6	36	141 Minislots	254 Bytes	719

First dynamic slot number is 65 for this reason message IDs are numbered from 65 to 79. First dynamic slot has the highest priority level and as the slot number increases priority level decreases. Experiment duration is 62 seconds. The results of the delay measurements for each message ID (dynamic slot number or priority level) are given in Table A-5. The maximum delay for each message is approximately 5000 µs which equals to one cycle length. It means that messages are never delayed because of insufficient dynamic segment length. Dynamic segment length in the configuration parameters are given as 2838 MT. Duration of a minislot is 6 MT. It means that one message is transmitted in 5 minislots which is also the dynamic slot duration. 15*5 = 75 minislots are enough to transmit all sporadic messages in one cycle. There are 472 minislots in the dynamic segment which means that all messages for each priority level can be sent in a single cycle. Time duration between two consecutive instants of a minislot is 5000 µs. For this reason average delay values are the half of this value which is 2500 µs. *DMR* = 0 for this experiment.

In the experiments 2 and 3 payload data length of the messages are first increased to 64 Bytes and than to 254 Bytes, which is the maximum allowed data length in

the protocol [13]. Detailed results for each dynamic slot ID (priority level) are given in Table A-6 and Table A-7 respectively. Dynamic slot durations are 24 minislots for 64 Bytes payload and 46 minislots for 254 Bytes payload. 360 minislots are enough to transmit 15 of 64 bytes messages in one cycle. For this reason the results of experiment 1 and 2 are similar. However 690 minislots are required to transmit 15 of 254 bytes messages in one cycle that means messages can be delayed for extra cycles because of insufficient dynamic segment length. However the generation times of the messages generally do not cause extra delays, there is only a slight increase for the maximum values. Average delay is again similar to experiments 1 and 2. Duration of the experiments is 63 and 115 seconds respectively. *DMR* = 0 for both experiments.

The comparison of the maximum and average delay values with respect to message priority levels for experiments 1, 2 and 3 are given in Figure 6-16.

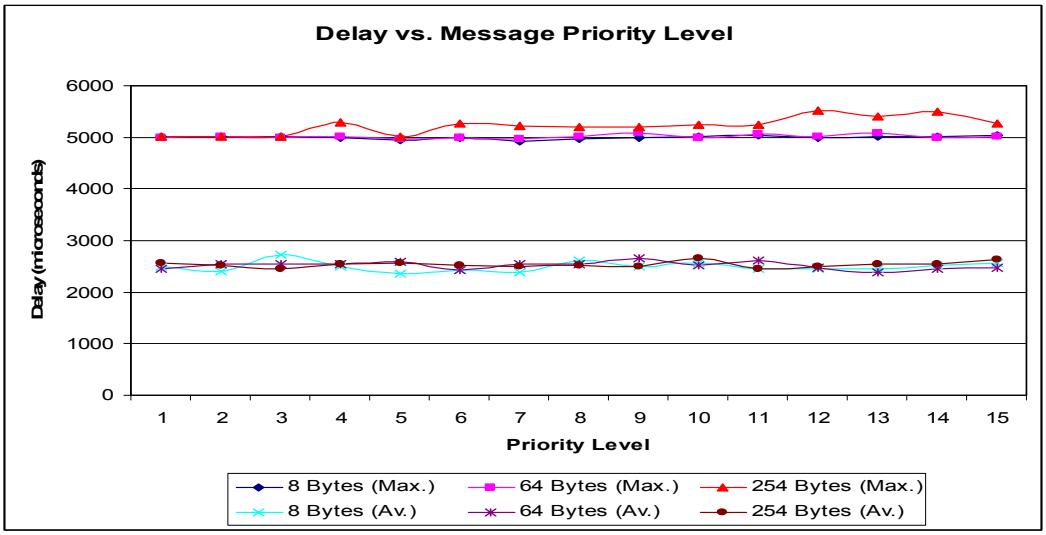

Figure 6-16: Maximum (µsec) & Average (µsec) Delays vs. Priority Levels: 15 Messages, 472 Minislots, - 8, 64 and 154 Bytes of Payload Data

In the next experiment (4) duration of the dynamic segment is decreased to 141 minislots by keeping all the other variables given in Experiment 3 the same. Some messages could not be sent in one or two cycles, since the available minislots are

less. Therefore observed maximum delay increased for low priority messages. However average delays are not affected in this case because of the low message traffic rate on the network. The results for this experiment are given in Table A-8. Duration of the experiment is 113 seconds. The comparison of maximum and average delay values with respect to priority levels for experiments 3 and 4 are given in Figure 6-17. *DMR* = 0 for Experiment 4.

Messages are generated periodically with exactly 50 ms periods in the next experiment (5) to increase the message traffic rate to 300 messages per second. In the previous experiments it was approximately 53 messages per second. This situation causes that lower priority messages within the same node experience more delays, since periods of the messages that belong to same node are equal and generation times of the different nodes are different than each other. The results for the delay calculations are given in Table A-9. The comparison of maximum and average delay values for experiments 4 and 5 with respect to priority levels are given in Figure 6-18. *DMR* = 0 for this experiment.

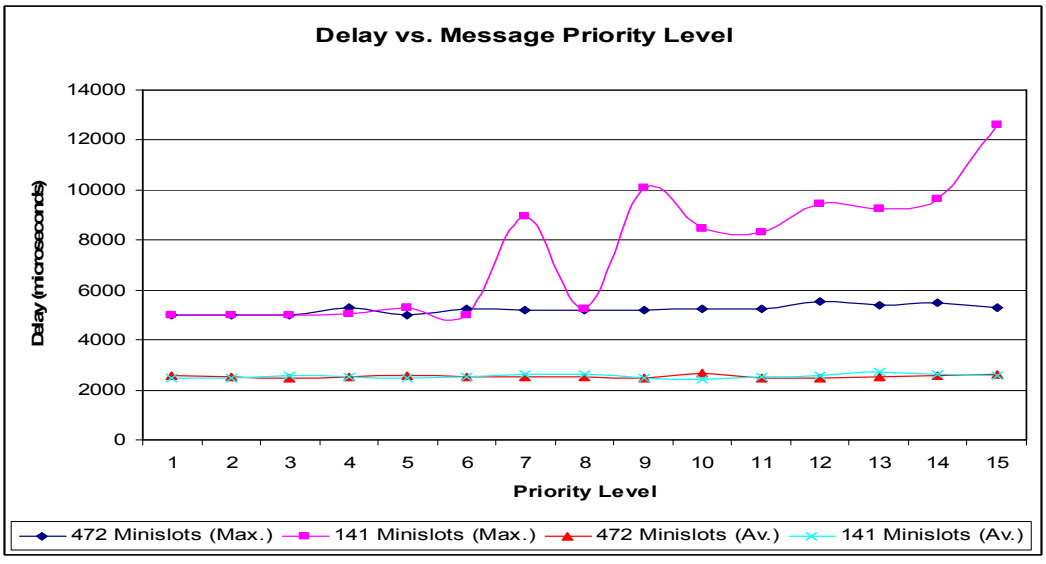

Figure 6-17: Maximum (µsec) & Average (µsec) Delays vs. Priority: 15 Messages, 254 Bytes of Payload Data - Dynamic segment length of 472 and 141 minislots

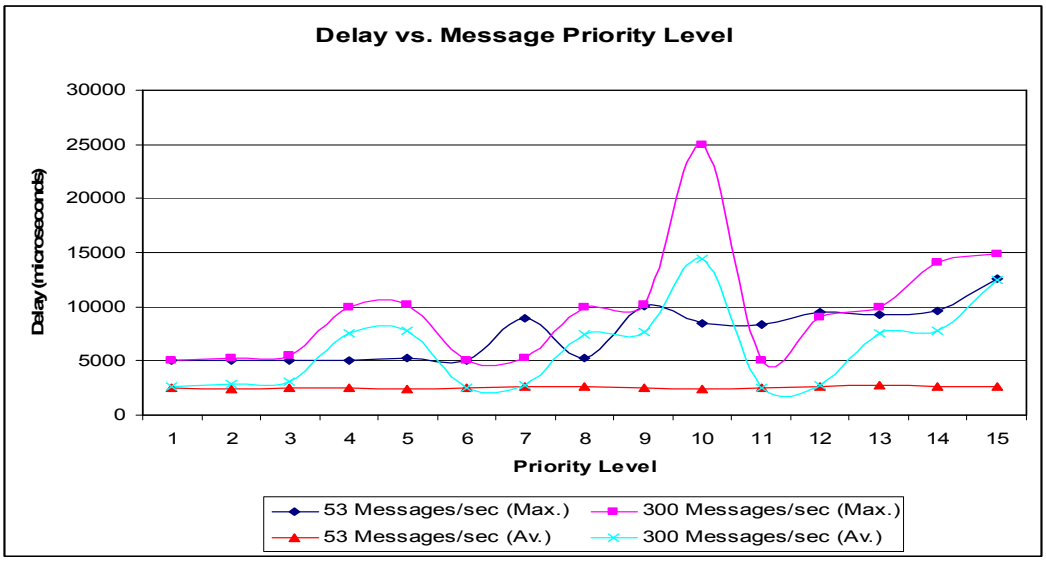

Figure 6-18: Maximum (µsec) & Average Delays (µsec) vs. Priority Level: 15 Messages, 254 Bytes of Payload Data, Dynamic segment length of 141 minislots - traffic rates of 53 messages/sec and 300 messages/sec

Evaluation of the dynamic segment with six nodes is performed with 36 sporadic messages. Each node generates 6 sporadic messages with 50 ms minimum interarrival time. Messages are scheduled starting from the first dynamic slot according to one message for one dynamic slot scheme as in 3 node experiments; only difference is the number of messages.

In the sixth experiment payload length of the sporadic messages is taken as 16 Bytes. Traffic rate during the experiment is 188 messages per second. 216 minislots are required to send all messages in one cycle. Since the available number of minislots is greater than required number of minislots maximum delays are around 5000 µs which is equal to one cycle duration. That means the only reason for the delay is timing hierarchy of the scheduling. Since the cycle length is 5000 µs average delays are around 2500 µs which is equal to half of cycle length. The results for the observed delays are given in Appendix I Table A-12. *DMR* is observed to be 0.

In the next two experiments (7, 8) payload length of the messages are increased to 64 and 254 Bytes. 504 and 1620 minislots are required to send all messages in a single cycle respectively. In these cases some messages may experience larger delays. That means if the generation times of large number of messages are in the same cycle lower priority messages are delayed to the next cycles because previous messages consume all available minislots. Maximum of 33 messages can be sent for 64 bytes payload and maximum of 10 messages can be sent for 254 bytes payload in a single cycle. Increasing the payload length causes the dynamic slots have higher durations and messages consume more minislots. That means available number of minislots decreases for the low priority messages and their dynamic slots are shifted to the end of cycle. Therefore there is a slight increase for the maximum delays. Because of low traffic load, average delays are approximately equal to the sixth experiment values. The computation results for the experiments are given in Table A-13 and Table A-14. Comparison of maximum and average delay figures of experiments 6, 7 and 8 for each priority level are given in Figure 6-19. *DMR* = 0 for both of the experiments.

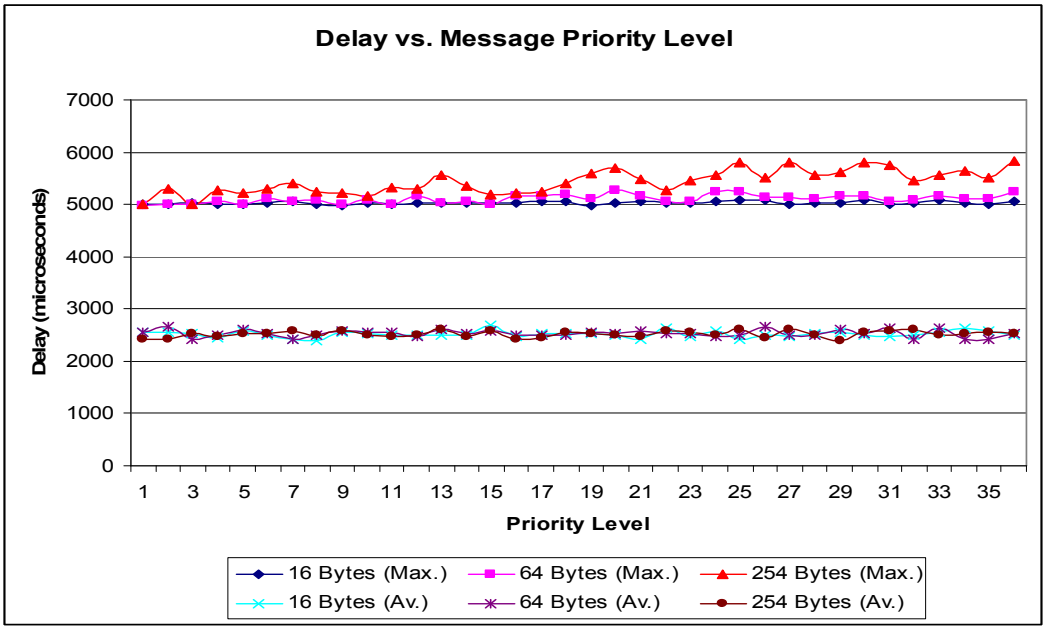

Figure 6-19: Maximum & Average Delays (µsec) vs. Priority Levels: 36 Messages, Dynamic Segment length of 472 minislots, 188 messages/sec – 16, 64 and 254 Bytes of payload data,

In the next experiment (9) payload length is fixed to 254 bytes and traffic rate is fixed to 188 messages per second. Length of the dynamic segment is decreased to 141 minislots. Each minislot has duration of 6 MT. Since available number of minislots is decreased, maximum delays experienced by the messages are increased significantly for lower priority messages. At most 3 messages can be transmitted in a single cycle. In the previous experiment this number is 10. Furthermore there is a slight increase for average values as expected. The computation results of the delay values for each message are given in Table A-15. Figure 6-20 shows the comparison of maximum and average delay values of experiment 8 and 9 with respect message priorities. *DMR* = 0 for the Experiment 9.

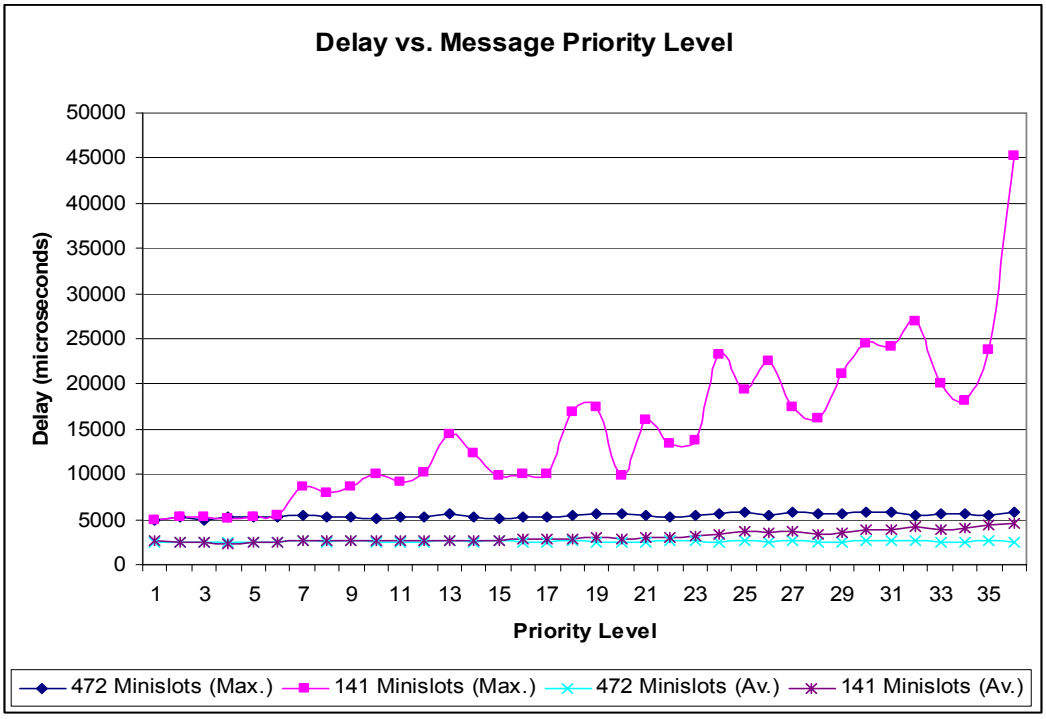

Figure 6-20: Maximum & Average Delays (µsec) vs. Priority: 36 Messages, 254 Bytes of payload data, 188 messages/sec - Dynamic Segment length of 472 and 141 minislots,

In the next experiment (10) traffic rate of transmitted messages is increased to 719 messages per second. The generated number of messages is actually more than this value, since some messages are can not be delivered because of the high traffic rate. In this case both maximum and average delay values show a considerable

increase. The nodes that possess the priority levels of 27 to 36 can never transmit any messages because of limited bandwidth and many messages for the priority levels of 19 to 26 miss their 50 ms deadlines. DMR is resulted to be 18815/44867 ≈ 0.42 by including the undelivered messages as deadline missed. The delay values for this experiment are given in Table A-16. Comparison of maximum and average delay values of the experiments 9 and 10 are shown in Figure 6-21 for each priority level.

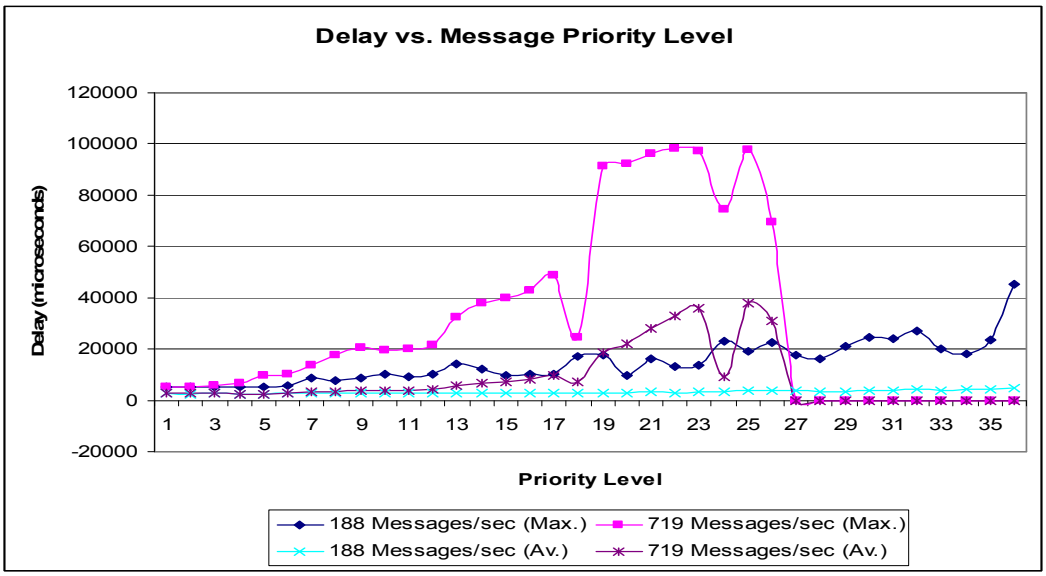

Figure 6-21: Maximum & Average Delays (µsec) vs. Priority - 6 Nodes, 254 Bytes of payload data, Dynamic Segment length of 141 minislots, 719 messages/sec and 188 messages/sec message traffic rate

6.2.2 Part 2 – Effect of Network Configuration Parameters

In this section, the effect of payload length of the messages, size of the dynamic segment, the number of message IDs, and message traffic load, on the delay of each message are examined. Furthermore the effect of priority of different payload lengths on delays under several dynamic segment lengths is examined. The experiments are divided into 6 categories and each above mentioned variable are changed in a controlled manner. After that delay values are calculated for each case. The control on the experiments is satisfied by keeping the other entire

variables constant and varying only the examined one. The configuration parameters that are kept constant through all experiments are given in Table 6-21. Other variables are given in the subsections. In the previous experiments duration of the dynamic segment is 472 minislots (2832 MT), in this section it is reduced to 307 minislots (1842 MT). The ID of the first dynamic slot is 97. Dynamic messages are scheduled starting from slot ID 97 and also message IDs are taken as equal to slot IDs. Therefore Message 97 has the highest priority. Priorities are numbered starting from 1 to number of messages. According to the values of configuration variables, the number of minislots that is used for a single physical frame transmission is calculated as follows. Dynamic slot duration for the messages with 16 Bytes of payload is 6 minislots, for the 64 Bytes of payload it is 14 minislots, for the 128 Bytes of payload it is 24 minislots and for the 254 bytes of payload it is 45 minislots.

All the experiments are performed three times to increase the reliability of the results and experiment duration in each performance is 90 seconds. Maximum delay values are the maximum of three experiments, average values and standard deviations are also calculated by considering three redundant experiments. *DMR* = 0 for this experiment.

Table 6-21: Configuration parameters that are kept constant

Configuration Parameter Name	Value
gdCycle (µs)	5000 µs
gPayloadLengthStatic (2-Byte-Words)	5 (10 Bytes)
gdActionPointOffset (MT)	5 MT
gdStaticSlot (MT)	31 MT
gNumberOfStaticSlots	96
Static Segment Length	96*31 = 2976 MT
gdMinislot (MT)	6 MT
gdMiniSlotActionPointOffset (MT)	2 MT
gdDynamicSlotIdlePhase (Minislot)	1 Minislots
gdNIT (MT)	178 MT
gdMacrotick (µs)	1 µs

6.2.2.1 Varying Size of Dynamic Segment

In this section, payload length, message traffic rate and the number of messages are kept constant and four different sizes for the dynamic segment are used. These values are given in Table 6-22. Since the static segment and cycle durations are fixed, if the dynamic segment length (DS) is decreased, NIT duration has to be increased to compensate this decrease. In the FlexConfig tool this decrease can be made without changing the NIT duration. The empty part (old DS – new DS) is undefined in the tool and this situation does not conform to the specifications given in [13]. In the implementation of the CC this unconformity does not cause any error.

Table 6-22: Varying Size of Dynamic Segment Experiments– Configuration Parameters

Variable	Value
Number of Message IDs	36
Payload Length of the dynamic messages	128 Bytes
Traffic Load	127 messages/sec
Number of Minislots	50 (0.3 ms), 100 (0.6 ms), 200 (1.2 ms), 307 (1.842 ms)

Since the required number of minislots to transmit a 128 Byte frame is 24 the last minislot number that a transmission can start is 26. For this reason only first 26 messages can be transmitted in the dynamic segment. This number is also the maximum number of messages that can be transmitted in a 50 minislots dynamic segment configuration with 128 bytes messages and given configuration parameters. For the 50 minislots case only two messages can be sent in a single cycle. For this reason delays increase as the priority of the message decrease. In other words the messages with high message IDs experience more delays and some messages missed their deadlines. For the 100, 200 and 307 minislot cases, all the 36 messages can be transmitted and delay values decrease as the number of

minislots is increased. The reason is that as the number of minislots is increased the available bandwidth for the transmission is increased. In these experiments no messages miss their deadline.

In the Figure 6-22 the comparison of maximum delay values for each message priority over four different dynamic segment length is given. Detailed calculation results of the message delays are given in APPENDIX I – Table A-17 to Table A-20. *DMR* is resulted to be 0.0002 for 50 Minislots case and zero for other cases.

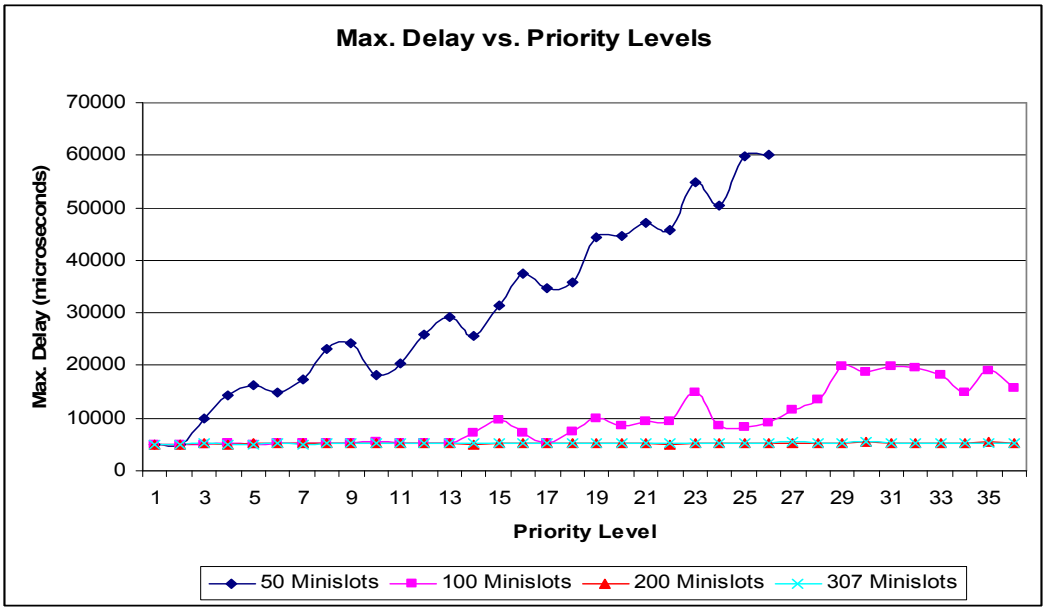

Figure 6-22: Maximum delay (µsec) vs. Priority: the case of varying dynamic segment length

Mean and standard deviation of delay values for each message priority are presented in Figure 6-23 and Figure 6-24. As the length of the dynamic segment is increased, mean and standard deviations decrease because of the increasing available number of minislots, that means increasing bandwidth.

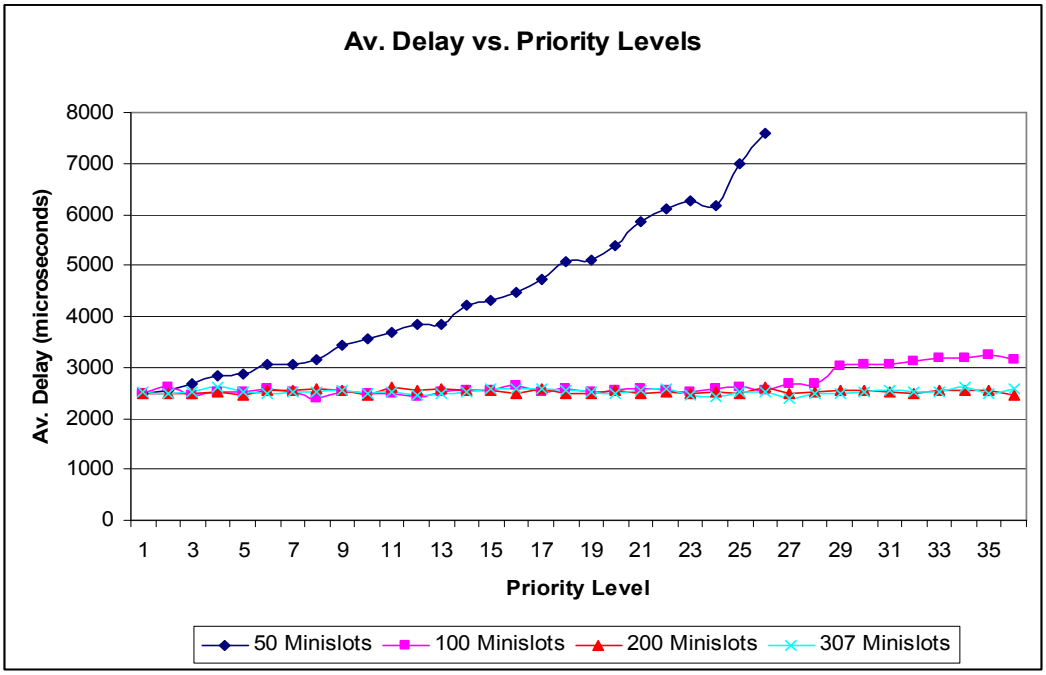

Figure 6-23: Average Delay (µsec) vs. Priority: the case of varying dynamic segment length

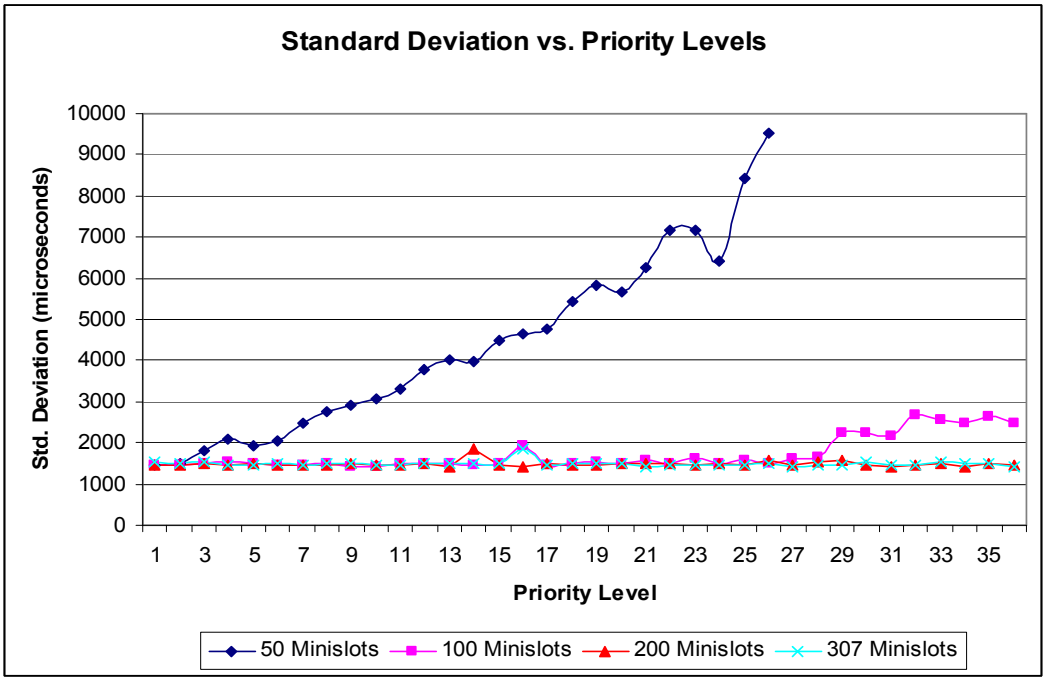

Figure 6-24: Standard Deviation (µsec) vs. Priority: the case of varying dynamic segment length

6.2.2.2 Varying Message Payload Length

In this part, number of messages, message traffic load and size of the dynamic segment is kept constant and the effect of payload length on delay values is examined. Values of the variables are given in Table 6-23.

Table 6-23: Varying message payload length experiments – Configuration Parameters

Parameters	Value
Number of Message IDs	36
Payload Length of the dynamic messages	16, 64, 128, 254 Bytes
Traffic Load	127 messages/sec
Number of Minislots	307 (1.842 ms)

Each message has the same length in all of the experiments. The effect of payload length on maximum delays for each priority level is shown in Figure 6-25. There is a slight increase as the payload length increases. Increasing payload lengths causes an increase in the duration of dynamic slots. Therefore the transmission times of lower priority messages shift to the end of dynamic segment if there are one or more transmissions in the previous dynamic slots. Since the traffic rate is not high as compared to FlexRay bandwidth, average delays and standard deviations are close to each other with different payload lengths. Mean and standard deviation of delays for each priority level are given in Figure 6-26 and Figure 6-27. $DMR = 0$ for this experiment. Detailed calculation results of the message delays are given in APPENDIX I – Table A-21 to Table A-24.

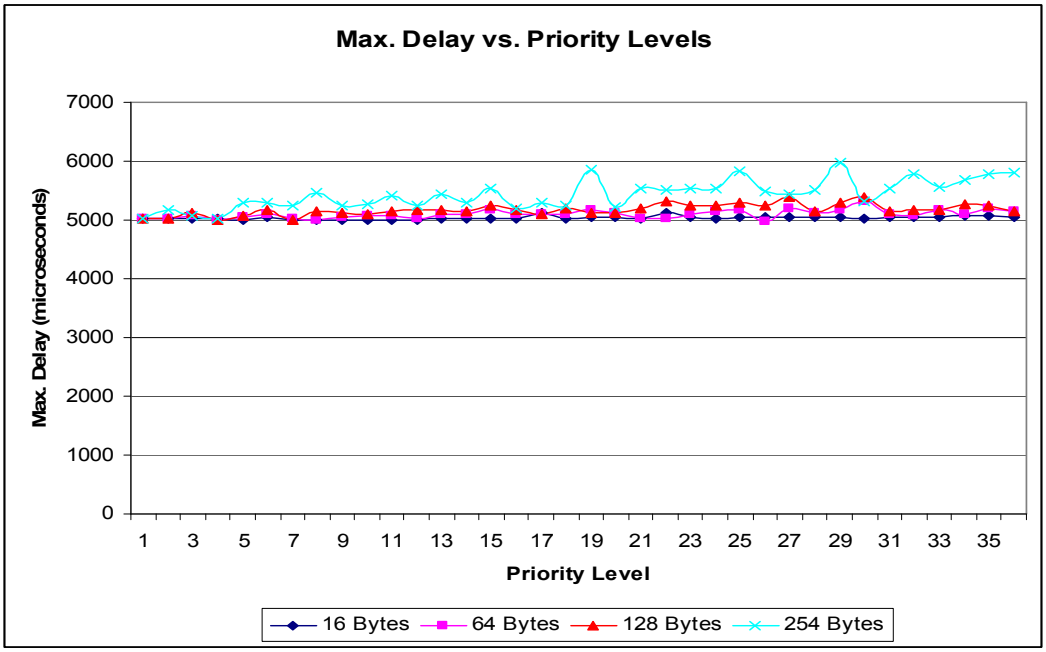

Figure 6-25: Maximum Delay (µsec) vs. Priority: the case of varying payload length

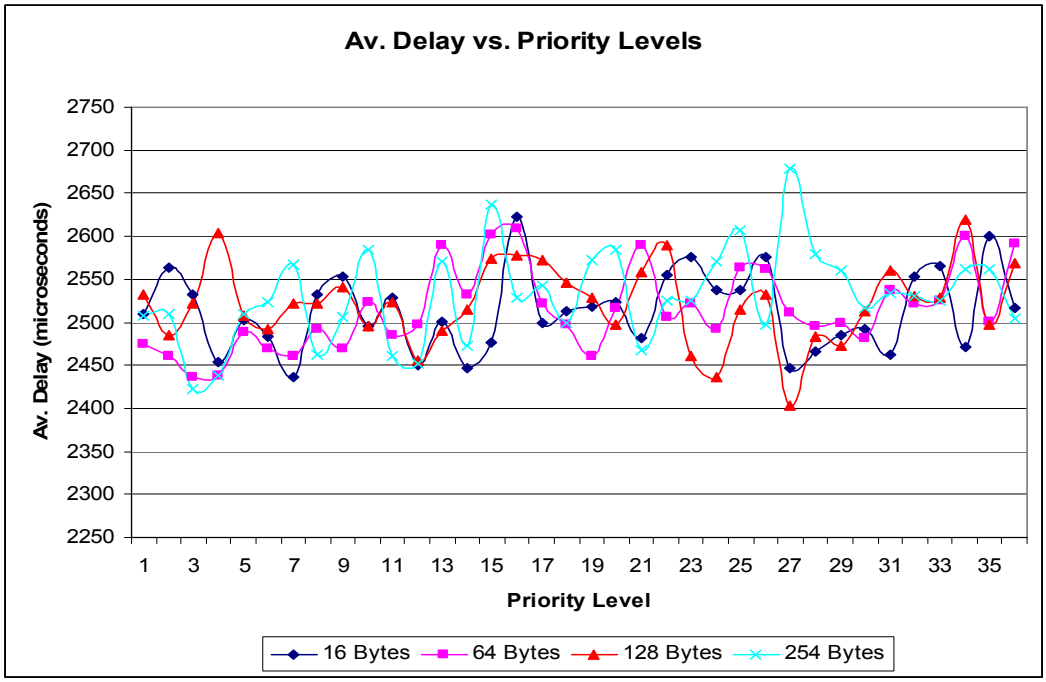

Figure 6-26: Average Delay (µsec) vs. Priority: the case of varying payload length

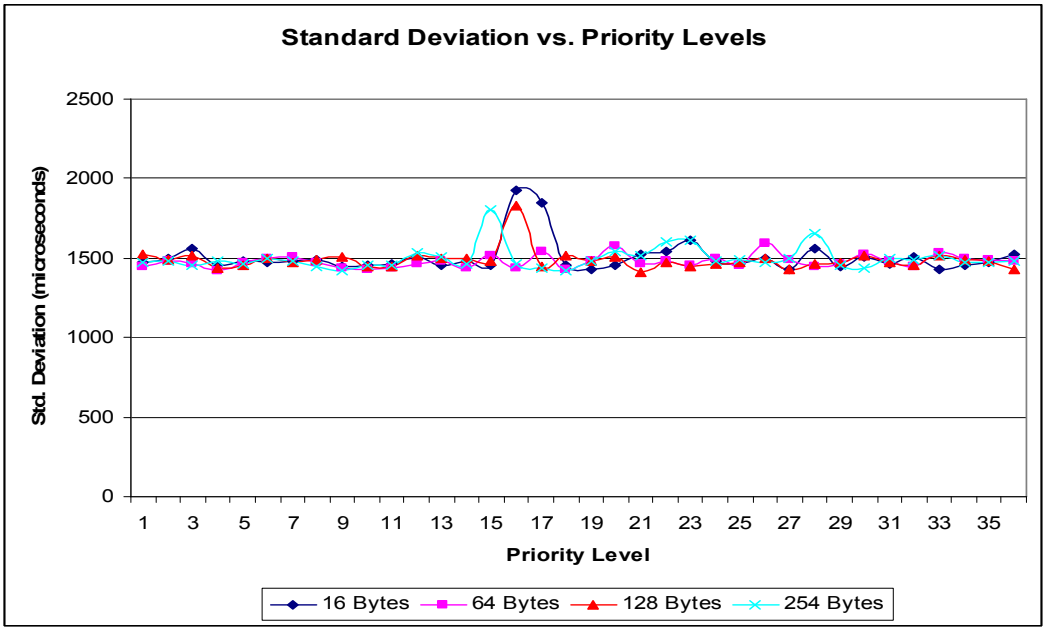

Figure 6-27: Standard Deviation (µsec) vs. Priority: the case of varying payload length

6.2.2.3 Varying Message Traffic Rate

Minimum interarrival times of the sporadic messages are set to be 50 ms for all experiments. Interarrival times are selected randomly between 50 ms and a given value to adjust the traffic load. This value is defined as the maximum allowed interarrival time. Because of the hardware constraint maximum interarrival time for the sporadic messages is 524 ms. Therefore, in the first experiment interarrival times is set to be random values between 50 ms and 524 ms. Then in the other experiments maximum value is reduced to 250 ms and then 100 ms to increase the message traffic load. In the run time traffic rates are observed as 127 messages/sec, 238 messages/sec and 480 messages/sec for 524, 250 and 100 ms maximum interarrival times respectively. Resulting configuration parameters are given in Table 6-24.

Table 6-24: Varying Message Traffic Load Experiments– Configuration Parameters

Parameters	Value
Number of Message IDs	36
Payload Length of the dynamic messages	128 Bytes
Traffic Load	127, 238, 480 messages/sec
Number of Minislots	307 (1.842 ms)

As the message load is increased the delay values increase for the low priority messages but this increase is relatively small because of the high bandwidth of the protocol. The probability of a previous message transmission increases and this probability is more for the lower priority messages. A previous message transmission means a shift on time of respective dynamic slot. Maximum, mean and standard deviation of the delay values are calculated and the detailed results are given in APPENDIX I – Table A-25 to Table A-27. Figure 6-28, Figure 6-29 and Figure 6-30 show the variations of Maximum observed delay values, mean and standard deviation of the message delays with respect to priorities. *DMR* = 0 for these experiments.

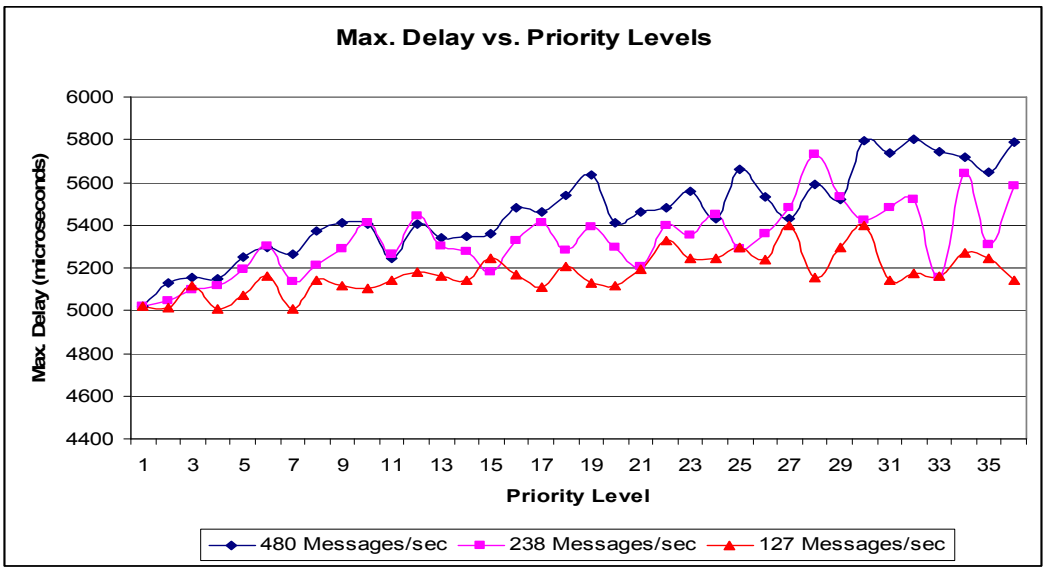

Figure 6-28: Maximum delay (µsec) vs. Priority: the case of varying message traffic load

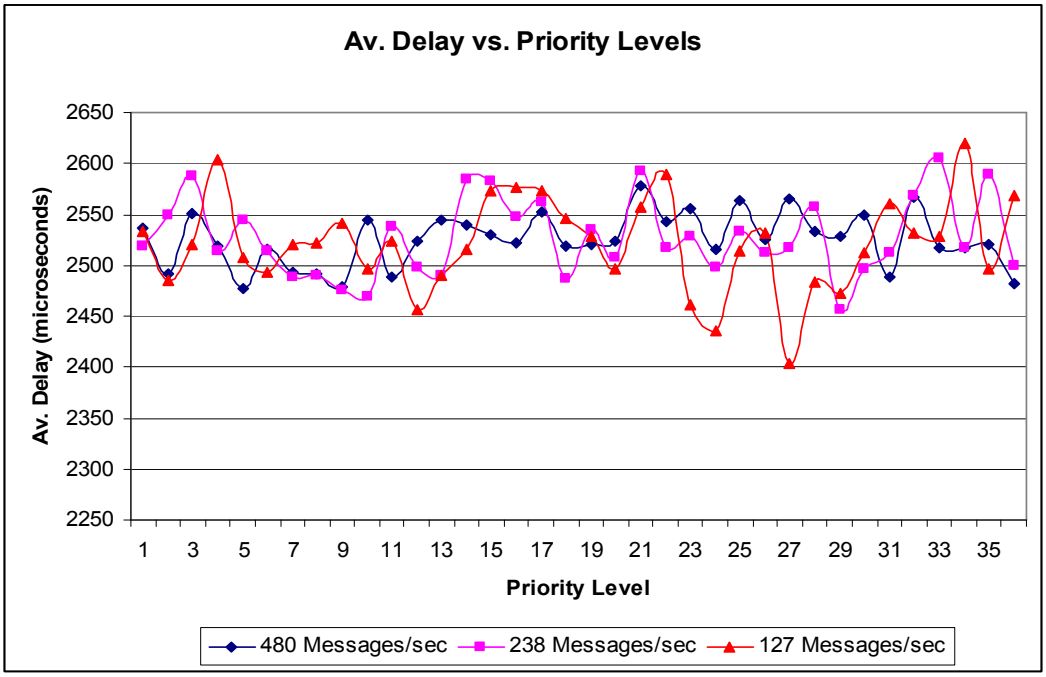

Figure 6-29: Average Delay (µsec) vs. Priority: the case of varying message traffic load

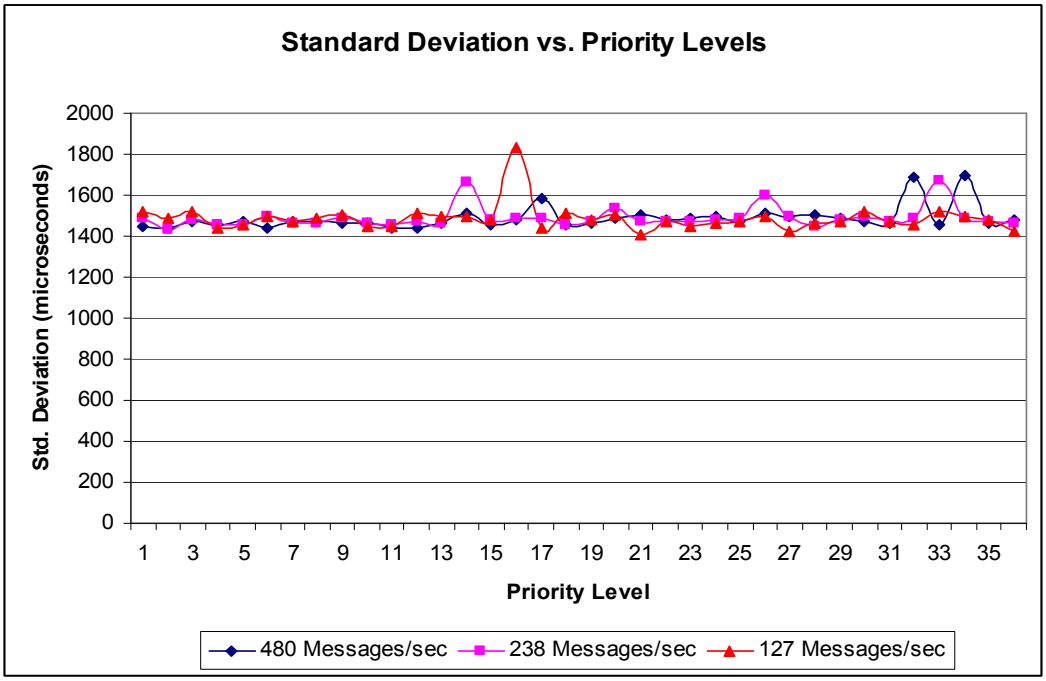

Figure 6-30: Standard Deviation (µsec) vs. Priority: the case of varying message traffic load

6.2.2.4 Varying the Number of Messages

In this part the number of messages is increased gradually from 36 to 84 messages and also this causes an increase in the message traffic rate. Interarrival times of newly added messages are set to be between 25 ms and 261 ms to increase the message traffic rate further and utilize the high bandwidth of the FlexRay protocol. Resulting configuration parameters are given in Table 6-25.

Table 6-25: Varying number of messages Experiments– Configuration Parameters

Parameters	Value
Number of Message IDs	36, 48, 66, 84
Payload Length of the dynamic messages	128 Bytes
Traffic Load	127, 224, 375 , 524 messages/sec
Number of Minislots	307 (1.842 ms)

The results show that there are not any noticeable increase in delay values and no missed messages even at a 522 messages/sec traffic load and 84 messages. High bandwidth of the FlexRay is enough to carry this message traffic without any degradation from the performance. However there is a small increase as the priority of the message IDs decreases. The reason for this increase is as the number of message ID increases the probability of a previous message transmission increases and this probability is more for the lower priority messages. A previous message transmission means a shift on time of respective dynamic slot. The calculation results of delay values are given in APPENDIX I – Table A-28 to Table A-31. Comparisons of maximum delays, mean and standard deviation of delays are given in Figure 6-31, Figure 6-32, and Figure 6-33. *DMR* = 0 for these experiments.

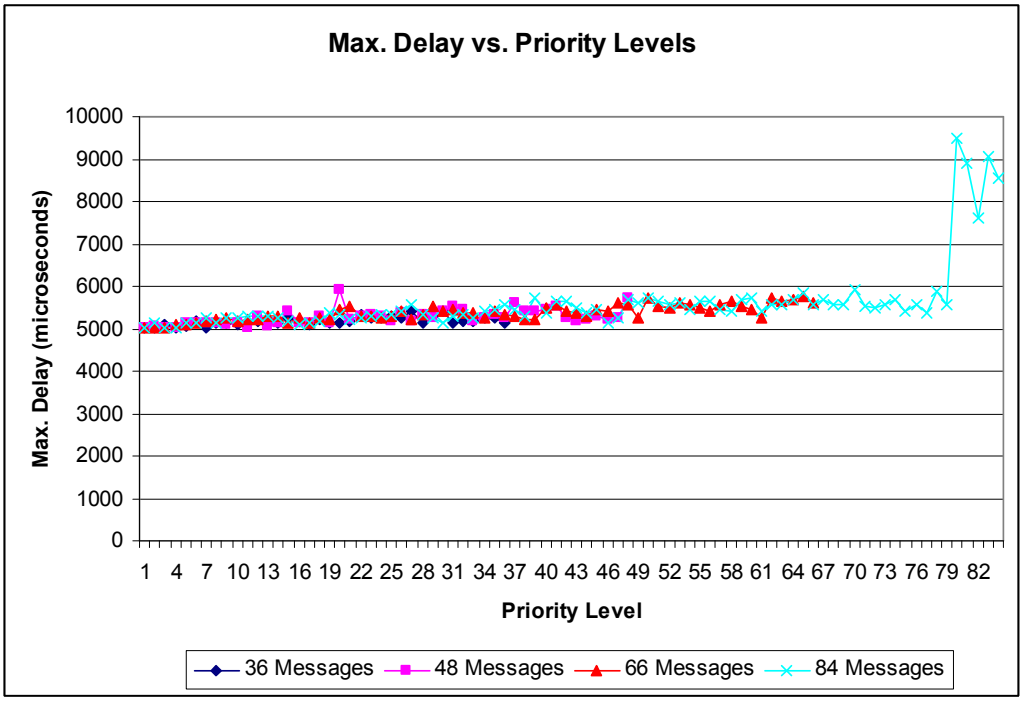

Figure 6-31: Maximum Delay (µsec) vs. Priority: the case of varying number of messages

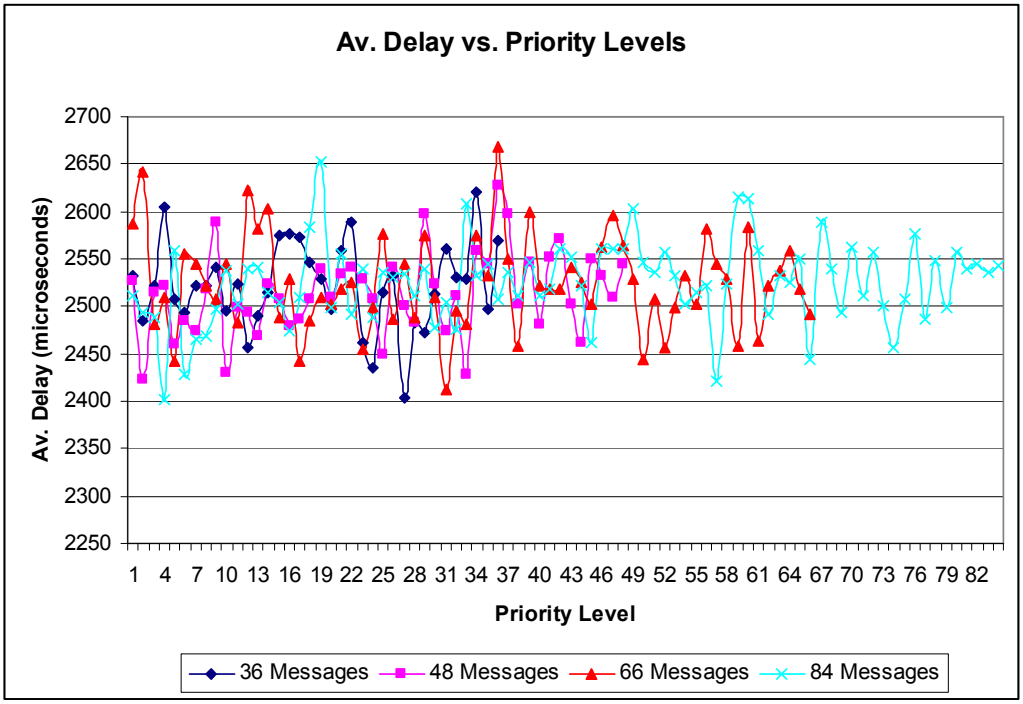

Figure 6-32: Average Delay (µsec) vs. Priority: the case of varying number of messages

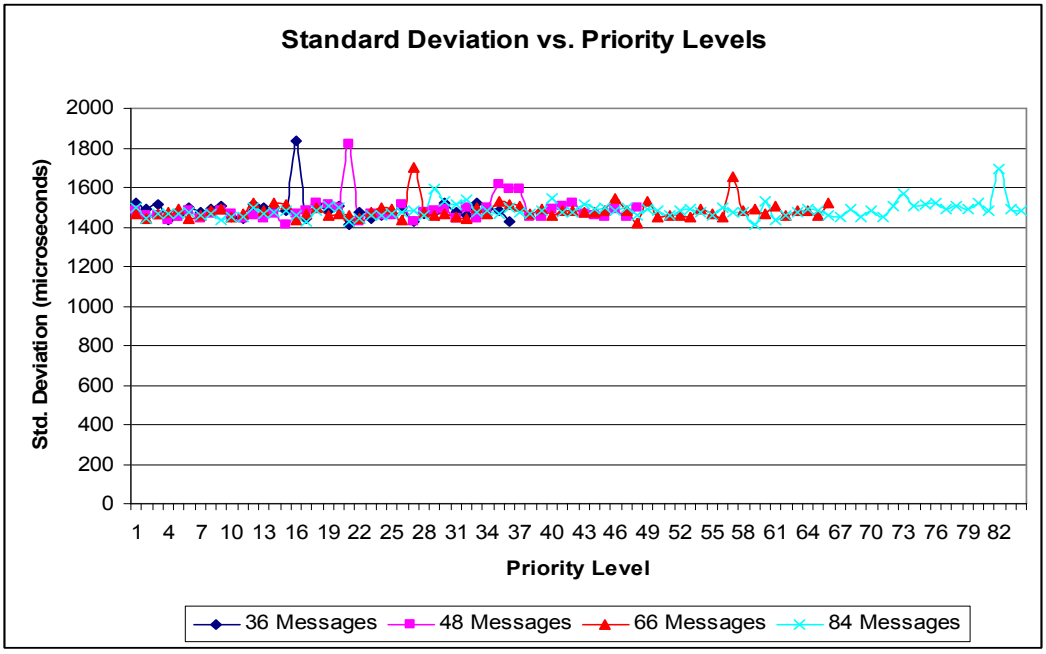

Figure 6-33: Standard Deviation (µsec) vs. Priority: the case of varying number of messages

6.2.2.5 Determining Priorities with respect to Payload Lengths for Several Dynamic Segment Durations

In the applications the payload length of the sporadic messages can be different. Up to now all messages has the same length in all experiments. In this case the scheduling method has an important affect on the delay values. Scheduling method in the dynamic segment can be defined as the deciding the priority levels of the messages IDs that have different payload lengths. In these experiments we investigate the effect of priority assignment which is according to payload lengths. Four different payload lengths are used for this purpose. 9 messages for each payload length are scheduled and experiments are repeated with three different dynamic segment lengths. Values of configuration parameters for this case are given in Table 6-26.

Table 6-26: Varying priorities with varying size of dynamic segment Experiments– Configuration Parameters

Parameters	Value
Number of Message IDs	36
Payload Length of the dynamic messages	16, 64, 128, 254 Bytes
Traffic Load	127 messages/sec
Number of Minislots	100 (0.6 ms), 200 (1.2 ms), 307 (1.842 ms)

In the first part 16 bytes messages scheduled with highest priorities and then 64, 128 and 254 byte messages assigned to minislots. Resulting scheduling scheme is given in the Figure 6-34.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
16	16	16	16	16	16	16	16	16	64	64	64	64	64	64	64	64	64
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
128	128	128	128	128	128	128	128	128	254	254	254	254	254	254	254	254	254

Figure 6-34: Scheduling Scheme – 16 Bytes messages have the highest priority

In the figure the upper rows with the numbers 1 to 36 show the priority of the message and lower rows shows the payload length. Effect of priorities is examined at dynamic segment lengths of 100, 200 and 307 minislots.

In the second part, selecting the priorities of the messages is reversed. 254 byte messages have the higher priorities then others. The scheduling scheme for this case is given in Figure 6-35.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
254	254	254	254	254	254	254	254	254	128	128	128	128	128	128	128	128	128
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
64	64	64	64	64	64	64	64	64	16	16	16	16	16	16	16	16	16

Figure 6-35: Scheduling Scheme – 254 Bytes messages have the highest priority

Again experiments are repeated with the dynamic segment length of 100, 200 and 307 minislots. Aim of these experiments is to see if scheduling long messages with higher priorities or scheduling short messages with higher priorities leads lower delay values. Furthermore, varying dynamic segment length leads to different results for both situations with the same message set.

The results of the experiments show that when the length of the dynamic segment is reduced, messages are experiencing more delays in both cases. Since as the dynamic segment length is decreased, the available bandwidth is reduced. For a longer dynamic segment length (307 minislots) the delay values are similar for each scheduling scheme. However as the allocated bandwidth for the dynamic segment is reduced the responses of each case are different. If the shorter length messages are scheduled with higher priority, then some of the 128 byte messages and 254 byte messages starting to experience more delays. These messages have lower priorities. As the length of the messages increases, the last minislot number that a transmission can start is decreasing. Therefore, the probability of insufficient number of minislots for the transmission increases when the reserved dynamic slot number for the message is high. The effect of previous message transmissions on the high payload length messages is noticeably high that means delay values of low priority messages are significantly high. However in the case of longer messages has higher priority the values of delays are different. Almost all the messages are

starting to experience higher delays, but as compared to former case these values are relatively small. This result shows that if the messages with high payload lengths scheduled with higher priorities, then all the other messages experiences more delays. However these delays are in the moderate levels and do not cause any message misses. If every message is to be transmitted with equal performance levels this scheduling strategy can be followed. If the of transmission is very important for some shorter payload length messages, and not as important for the longer length messages then second scheduling strategy will be a better choice. The comparison of maximum delay values and mean and standard deviation of each transmitted messages with respect to priorities are given in Figure 6-36 through Figure 6-41. In the figures, the word "decreasing" is used for the case of long payload length messages scheduled with higher priorities, and "increasing" is used for the other case. The detailed results of delay calculations are given in APPENDIX I – Table A-32 to Table A-37. *DMR* = 0 for all experiments in this section.

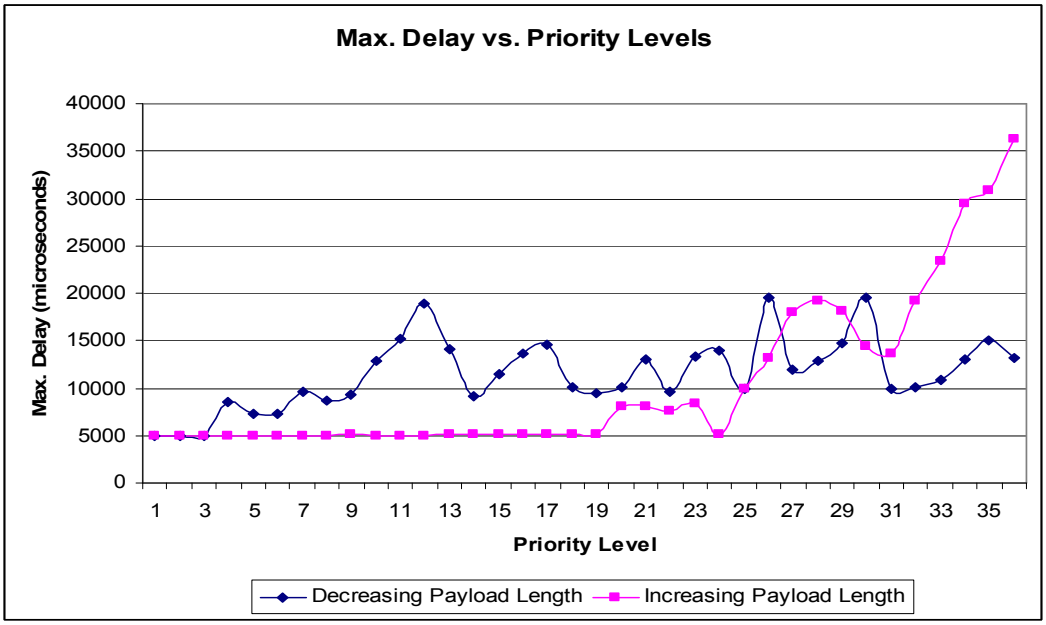

Figure 6-36: Maximum Delay (µsec) vs. Priority: the case of 100 minislots of Dynamic segment

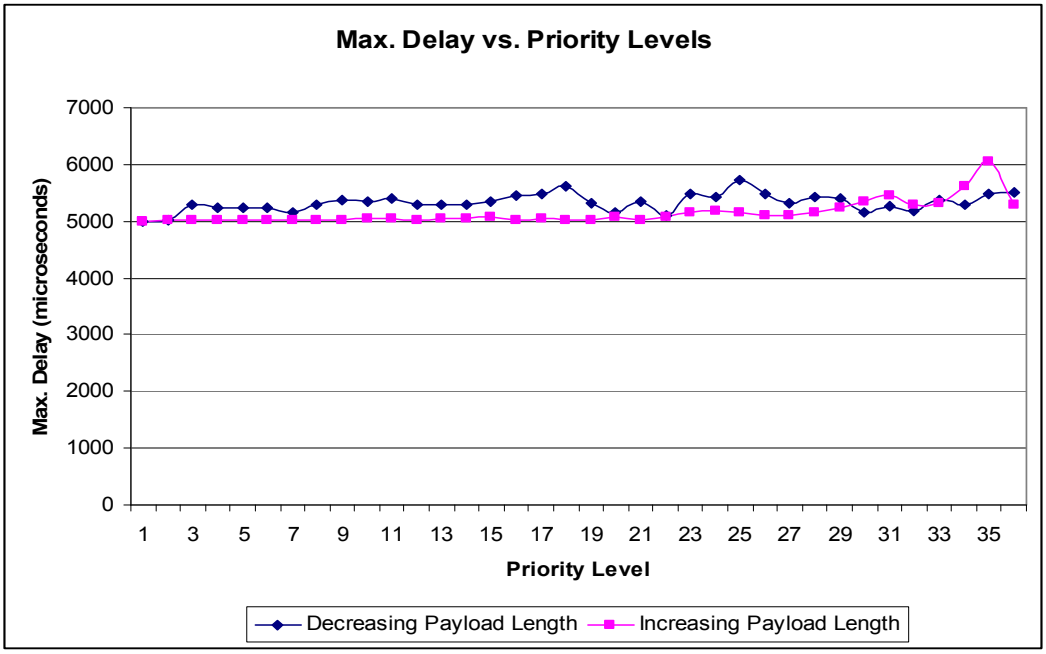

Figure 6-37: Maximum Delay (µsec) vs. Priority: the case of 200 minislots of Dynamic segment

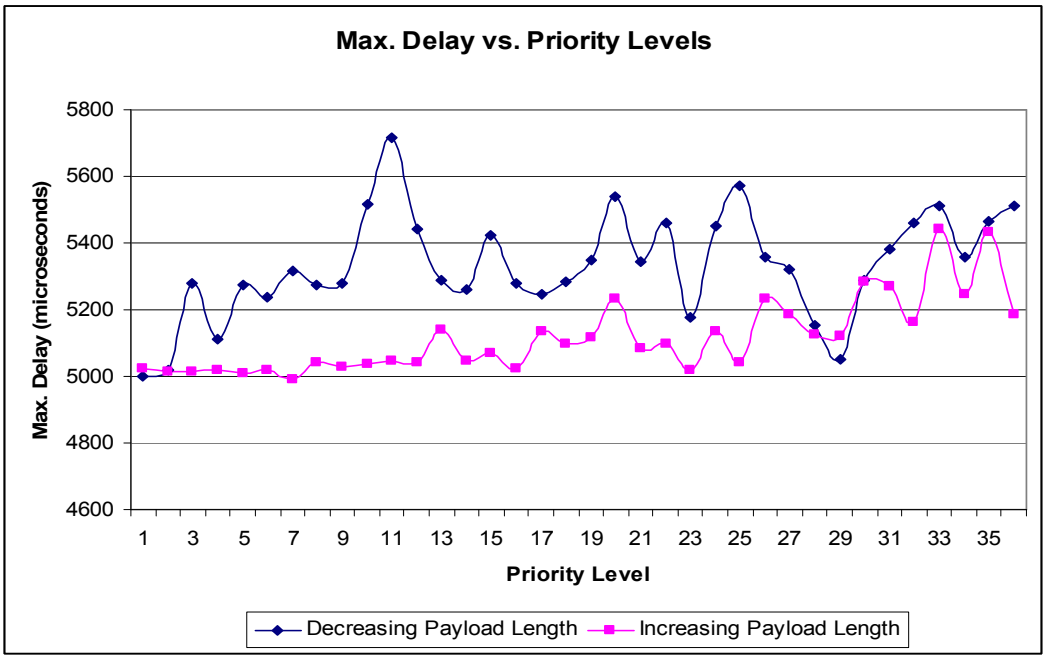

Figure 6-38: Maximum Delay (µsec) vs. Priority: the case of 307 minislots of Dynamic segment

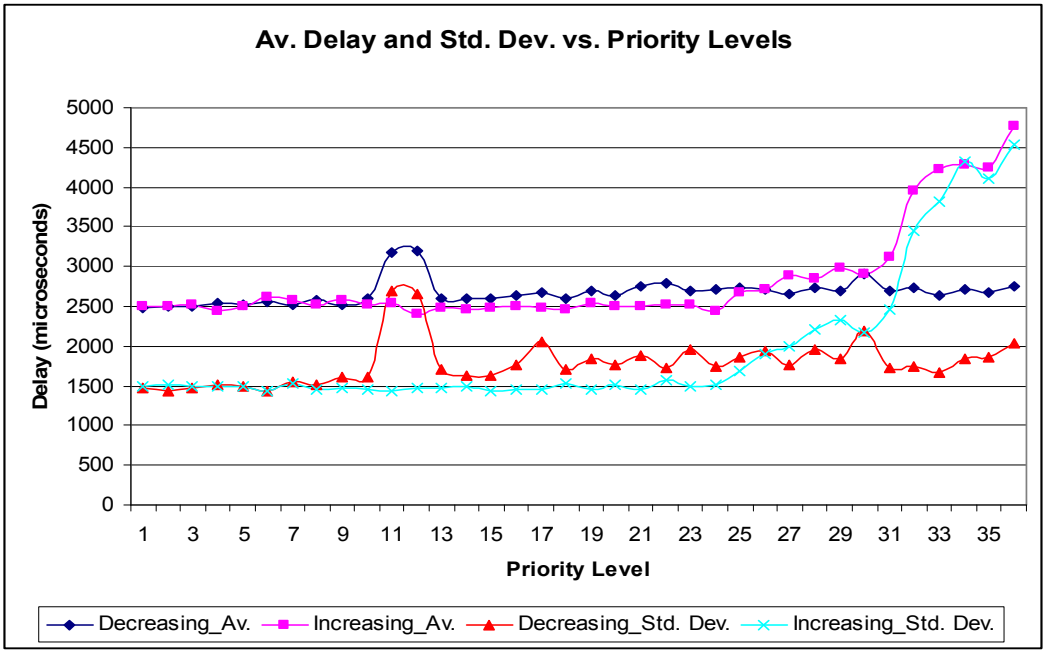

Figure 6-39: Average Delay & Standard Deviation (μsec) vs. Priority: the case of 100 minislots of Dynamic Segment

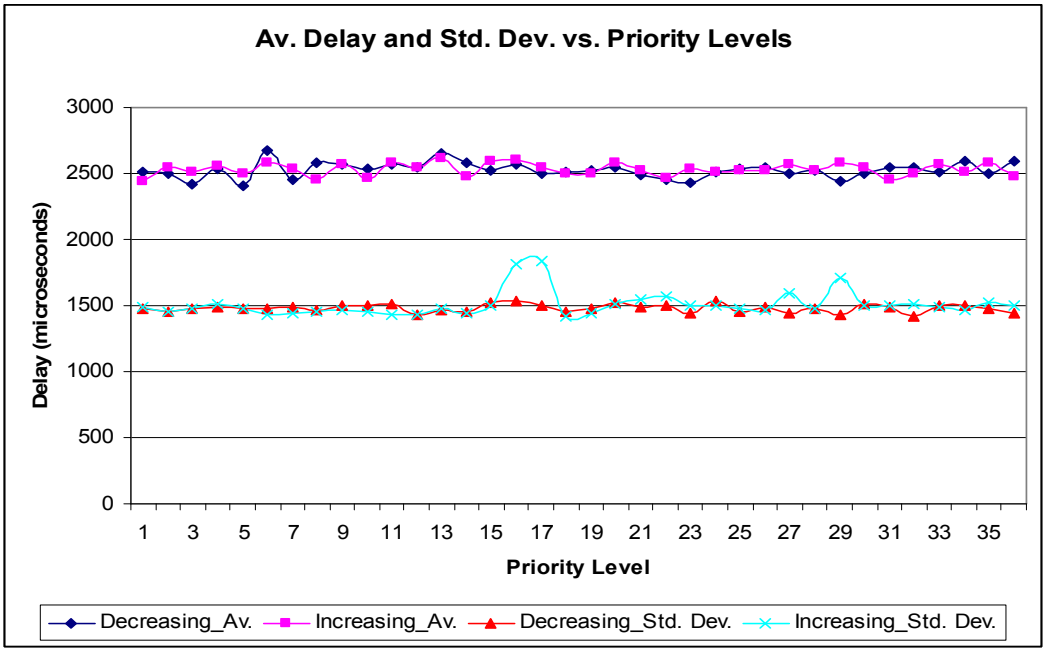

Figure 6-40: Average Delay & Standard Deviation (μsec) vs. Priority: the case of 200 minislots of Dynamic Segment

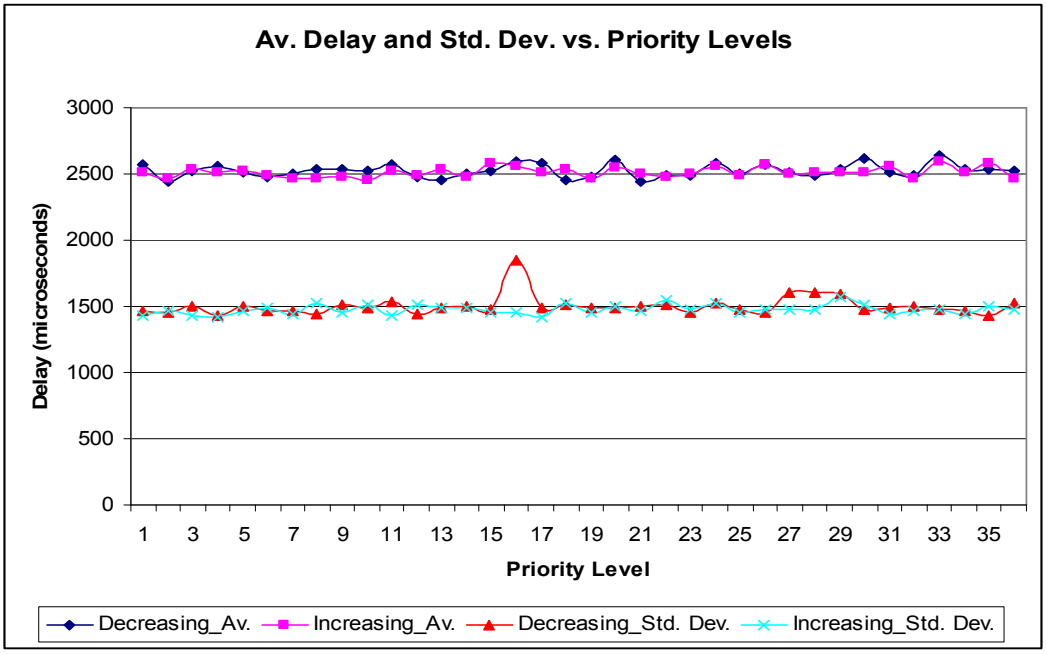

Figure 6-41: Average Delay & Standard Deviation (µsec) vs. Priority: the case of 307 minislots of Dynamic Segment

6.2.3 Summary and Discussion of the Results of the Dynamic Segment Experiments

Dynamic segment experiments are performed in two steps. In the first step variations of the delay values for the messages with respect to message payload length, dynamic segment length, and message traffic rate and number of messages are explored. The results show that as the ratio of (*Message Traffic Rate*)/(*Dynamic Segment Length*) or (*Payload Length*)/(*Dynamic Segment Length*) are increased delay values for the low priority messages (messages with high ID) increase. Minimum value for the Maximum delay is around 5 ms which is the cycle length. These values can be decreased by choosing shorter cycle length duration. However in this case available number of minislots in one cycle will be reduced. In 5 ms cycle duration case, dynamic segment duration is 1.842 ms, when cycle length is reduced to 2.5 ms keeping the same dynamic segment length/cycle length ratio dynamic segment length will be around 0.9 ms.

In the second part of the experiments effect of each variable (message traffic rate, dynamic segment length, and number of messages) is examined individually. The delay values of the messages start to increase only at extreme levels of these variables. These extreme levels are 50 and 100 number of minislots dynamic segment length, 254 Byte message Payload Length, 480 messages/sec traffic rate and 84 sporadic messages. Furthermore the deadlines are missed for some of the messages under situations that the combination of some of the limit values for variables are used. Apart from the combinations of limit values FlexRay dynamic segment can continue to the message transmission with acceptable levels for delay values. This situation makes the dynamic segment very suitable for sporadic messages for the given configuration parameters because of the structure and high bandwidth of the FlexRay.

CHAPTER 7

CONCLUSIONS

In this thesis, performance of the FlexRay networks for in-vehicle communication is experimentally evaluated and the results of the real time implementations are presented. All of the experiments are realized in a real time hardware environment. Commercially available electronic units are used for the implementations. Both the Static and the Dynamic Segments of the protocol are examined in the thesis. The structure of the protocol allows independent evaluation of these segments. The reason for experimental evaluation is to see the differences from a simulation. Nonzero minimum jitter and realistic static and dynamic durations including the physical media can be given as examples for these differences. Furthermore in the experiments, clock and rate corrections are performed by nodes and physical frame encodings leads to a realistic evaluation.

FlexRay is a newly developed protocol and first examples of its usage in the mass production automobiles can be seen on the market. However satisfactory information about its real time performance is not available in the literature yet. In this thesis, commercially available FlexRay products are used to implement a real FlexRay environment, configure the network and analyze the message transmission on the network. These products allow custom made transmission scenarios to be implemented and tracking the network traffic. Network traffic then can be recorded and the timing properties of each message transmission can be examined.

In this work we first define performance metrics for the message transmission on the FlexRay Networks followed by an experimental study to evaluate these metrics.

We define message *delay* and *jitter* as performance metrics that measure the quality of service received by the periodic messages transmitted in the FlexRay static segment. In this respect, number of sporadic messages that miss their deadlines is measured by the *deadline miss ratio* metric. We also measure how efficiently the network bandwidth is used by *utilization*, *used static slot IDs*, *static slot allocation* metrics. The examination results of each transmission scenario are later evaluated and the values of the defined performance metrics are computed. Values for the performance metrics are compared for related scenarios.

The implementation of the communication scenarios and also verification of the Evaluation Board, which is used as the main hardware are carried out on a small scale network of 3 nodes. The experiments are then repeated in a network of 6 nodes

Two kinds of messages are used in the evaluation. These are periodic messages transmitted in the Static Segment and sporadic messages transmitted in the dynamic segment. In the large scale network there are 51 periodic messages for 6 nodes.

In the static segment experiments, the message scheduling is performed starting from the simplest scheduling scheme and then with the methods given in [21] by using message packing, cycle filtering, and cycle multiplexing the utilization is increased, Static Slot ID usage and Static Slot Allocation is decreased.. The simplest scheduling method, which is allocating one static slot in every cycle for each message, leads to the use of 51 static slots and 28% utilization. Then utilization is increased to 66% with cycle filtering. By this way additional messages can be added to the network without using extra static slot. Then cycle multiplexing is applied and static slot usage is reduced to 21 without effecting the utilization. By this way additional ECUs can be added to the network without the need for extra bandwidth. Then message packing is applied to further decrease U_{ID} and *SA* and increase utilization. All the above mentioned work is done while jitter is kept at minimum. Then allowing jitter in the transmission times of the messages

static slot usage is reduced to 15 and utilization is increased to 97.3%. The deadlines of the messages are met in all experiments and no message loss is observed. The structure of the static segment guarantees that all messages scheduled in this segment are transmitted on time. In the case that message transmissions without jitter only the 10.8% of the bandwidth of the network is used and a total of 51 messages with 10 bytes of message data are transmitted.

For sporadic message transmissions dynamic segment is used and experiments are designed to see the effect of message payload data, dynamic segment length, message traffic load, priority, and number of messages. In the first set of experiments message payload data are gradually increased to 254 bytes which is the maximum payload length allowed for a FlexRay frame. Then length of the dynamic segment is reduced and finally message traffic load increased. This sequence of experiments is applied to increase the delays as much as possible for the stress testing of the FlexRay network. Almost all the messages are transmitted without any error and no deadlines are missed except some messages missed their deadlines in extreme conditions which are very limited dynamic segment length and very high message load with 254 bytes of payload data. In the second set of experiments the effect of each variables are examined one by one. The behavior of delay values for each network configuration is explored and outcome of the methods for the schedule of a message set with several payload data lengths is presented. The results show that as the ratio of (*Message Traffic Rate*)/(*Dynamic Segment Length*) or (*Payload Length*)/(*Dynamic Segment Length*) are increased delay values for the low priority messages (messages with high ID) increase. For the case of several messages with several payload lengths, scheduling the longer messages with high priority causes all the other messages experiencing large delays. However, if the messages with shorter payload lengths are scheduled with higher priorities only messages with longest payload length (lowest priority messages) experience high delay.

X-by-Wire applications demand for signals to be transmitted on time. The results of the thesis show that strict timing constraints of these applications can be satisfied

by FlexRay. Static segment can be used for the transmission of safety critical periodic messages because of its determined, stable and configurable structure. Furthermore dynamic segment of the protocol is very suitable for non-safety critical event driven applications. The result of the experiments shows the conformity of the FlexRay protocol to expectations.

This thesis covers only the performance of the FlexRay network in fault free conditions. As a future work communication scenarios can be expanded to include faulty conditions such as EMI exposure from the outside world and internal fault injection. Static segment length can be reduced to save more bandwidth by decreasing static slot action point offset. However, effect of this decrease on the transmission errors such as boundary violation should be carefully examined both under fault free and faulty conditions. Other parameters of the network can be examined to see their effects on delay values for dynamic segment messages. For example total traffic rate of the messages together with number of message IDs can be increased to heavily load the network, because using sensor networks integrated with FlexRay can require mass amount of different type signal transmission. Minislot length can be an important parameter in the case of very large number of message IDs. Furthermore different scheduling techniques apart from adjusting the protocol parameters such as allocating more dynamic slots to the message IDs with different priorities can be used to decrease delay values. In our experiments we have used a simple bus topology. However to increase the safety and reliability of the system different bus topologies can be used. The performance of the network can be examined under different bus topologies and advantages or disadvantages of each topology can be presented. By this way, performance of the protocol under faulty conditions and the effect of the internal properties of protocol and external affects can be examined.

REFERENCES

[1] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE,* vol.93, no.6, pp. 1204-1223, June 2005.

[2] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of Controller Area Network," Department of Signals, Sensors and Systems, Royal Institute of technology, Stockholm, Sweden; Department of Electrical Engineering, Linkoping University, Sweden, Technical Report, 2003.

[3] G. Leen, D. Heffernan, "Expanding automotive electronic systems," *IEE Proc.-Comput. Digit. Tech,* vol.35, No.1, pp. 88-93, January 2002.

[4] J. Bell, "Network protocols used in the automotive industry," The University of Wales, Aberystwyth, June 2002.
Available: www.aber.ac.uk, (accessed December 2009).

[5] G. Cena, A. Valenzano, and S. Vitturi, "Advances in automotive digital communications," *Computer Standards and Interfaces 27 (2005),* pp. 665-678, January 2005. Available: http://www.sciencedirect.com (accessed December 2009).

[6] Freescale Semiconductor Inc, "Controller Area Network.". Available: http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=02205025B4, (accessed December 2009).

[7] Local Interconnect Network Specification, LIN Consortium. Available: http://www.lin-subbus.org/, (accessed December 2009).

[8] T. Nolte, H. Hansson, and L.L. Bello, "Automotive communications – past, current and future," *10th IEEE Conference on Emerging Technologies and Factory Automation,* pp. 985-992, September 2005.

[9] P. Pop, P. Eles, and Z. Peng, "Analysis and optimization of heterogeneous real-time embedded systems," *IEE Proc.-Comput. Digit. Tech,* vol. 152, No. 2, pp. 130-147, March 2005.

[10] J. Broy, K.D. Müller-Glaser, "The impact of time-triggered communication in automotive embedded systems," SIES '07. presented at International Symposium on Industrial Embedded Systems, Lisbon, July 4-6, 2007.

[11] A. Karlsson, "X-by-wire systems and time-triggered protocols," M.S. thesis, Dept. of Information Technology, Uppsala University, Sweden, 2002.

[12] R. Makowitz, C. Temple, "FlexRay – A communication Network for Automotive Control Systems," *International Workshop on Factory Communication Systems,* pp. 207-212, June 2006.

[13] The FlexRay Communications System Specifications, Ver. 2.1 Available: www.flexray.com, (accessed December 2009).

[14] TTTech Computertechnik. Available: www.tttech.com, (accessed December 2009).

[15] J. Berwanger, M. Peller, and R. Griessbach, "Byteflight – A new protocol for safety critical applications," Presented at FISITA World Automotive Congress, Seoul, Korea, June 12-15, 2000.

[16] BMW website. Available: www.bmw.com, (accessed December 2009).

[17] Software Guide for SK-91465X-100MPC, MB91460 Series Evaluation Board. Available: http://www.fujitsu.com/emea/, (accessed December 2009).

[18] SK-91465X-100MPC Fujitsu Evaluation Board documentation. Available: http://mcu.emea.fujitsu.com/mcu_tool/detail/SK-91465X-100PMC.htm, (accessed December 2009).

[19] Fujitsu FlexRay Driver User Manual, Reference is only available within the SK-91465X-100MPC Fujitsu Evaluation Board Package.

[20] Eberspaecher Electronics website. Available: http://www.eberspaecher.com/servlet/PB/menu/1053178_l2/index.html, (accessed December 2009).

[21] K. Schmidt, E.G. Schmidt, "Message scheduling for the FlexRay protocol: The Static Segment," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 2170-2179, June 2009.

[22] M. Kang, K. Park, and B. Kim, "Determining the size of a static segment and analyzing the utilization of in-vehicle FlexRay network," *Third International Conference on Convergence and Hybrid Information Technology,* vol. 2, pp. 50-53, November 2008.

[23] Dr. Anton Schedl, "Goals and architecture of FlexRay at BMW," Presented at Vector FlexRay Symposium, Stuttgart, Deutschland, March 2007.

[24] N. Navet, F. Simonot-Lion, "The Automotive Embedded Systems Handbook," *Industrial Information Technology series, CRC Press / Taylor and Francis*, ISBN 978-0849380266, 2008.

[25] G. Leen, D. Heffernan, A. Dunne, "Digital networks in the automotive vehicle," *Comput. Control Eng. J.,*vol.10, pp.257-266, December 1999.

[26] E.G. Schmidt, K Schmidt, "Message scheduling for the FlexRay protocol: The Dynamic Segment" *IEEE Transactions on Vehicular Technology*, vol.58, pp.2160-2169, Jun 2009.

[27] E.A. Bretz, "By-wire cars turn the corner" *IEEE Spectrum*, vol.38, pp.68-73, April 2001.

APPENDIX A

Table A-1: Static Segment, Experiment 1: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (µs)	Max. Delay (µs)	Av. Delay (µs)	Min. Jitter (µs)	Max. Jitter (µs)	Av. Jitter (µs)
Message 1	6182	1234	1308	1271	1	2	1.2
Message 2	12364	1248	1338	1293	0	1	0.6
Message 3	3091	1154	1228	1191	2	3	2.4
Message 4	6182	1261	1352	1306	1	2	1.2
Message 5	6182	1296	1370	1333	1	2	1.2
Message 6	6182	1169	1243	1206	1	2	1.2
Message 7	6182	54	129	92	1	2	1.2
Message 8	6182	60	134	97	1	2	1.2
Message 9	6182	97	188	142	1	2	1.2
Message 10	6182	128	202	165	1	2	1.2
Message 11	6182	141	232	186	1	2	1.2
Message 12	3091	172	246	209	2	3	2.4
Message 13	6182	203	277	240	1	2	1.2
Message 14	3091	216	290	253	2	3	2.4
Message 15	6182	230	321	275	1	2	1.2
Message 16	6182	306	381	344	1	2	1.2
Message 17	6182	312	386	349	1	2	1.2
Message 18	6182	341	416	379	1	2	1.2
Message 19	618	360	419	389	11	13	13
Message 20	1236	377	450	414	4	7	6
Message 21	619	416	489	453	11	13	13
Message 22	618	429	503	466	11	13	13
Message 23	618	477	551	514	11	13	13
Message 24	248	618	692	655	29	31	31
Message 25	124	649	722	687	59	61	61
Message 26	247	563	636	599	29	31	31
Message 27	6182	567	675	624	1	2	1.2
Message 28	618	615	671	642	11	13	13
Message 29	618	638	724	689	11	13	13
Message 30	618	685	759	728	11	13	13
Message 31	31	718	737	727	240	243	243
Message 32	31	720	758	730	240	243	243
Message 33	62	734	753	744	120	122	122
Message 34	62	808	882	837	120	122	122
Message 35	3091	932	1032	993	2	3	2.4
Message 36	31	853	890	880	240	243	243
Message 37	31	923	943	934	240	243	243
Message 38	30	908	978	945	240	243	243
Message 39	31	961	1021	993	240	243	243
Message 40	31	1122	1182	1152	240	243	243
Message 41	618	1011	1070	1040	11	13	13

Table A-2: Static Segment Experiment 2: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (µs)	Max. Delay (µs)	Av. Delay (µs)	Min. Jitter (µs)	Max. Jitter (µs)	Av. Jitter (µs)
Message 1	6233	6234	6308	6271	1	2	1.2
Message 2	12465	1248	1338	1293	0	1	0.6
Message 3	3117	1154	1228	1191	2	3	2.4
Message 4	6233	1261	1352	1306	1	2	1.2
Message 5	6233	1296	1370	1333	1	2	1.2
Message 6	6233	6169	6243	6205	1	2	1.2
Message 7	6233	5054	5129	5092	1	2	1.2
Message 8	6233	60	134	97	1	2	1.2
Message 9	6233	96	188	142	1	2	1.2
Message 10	6233	5128	5202	5165	1	2	1.2
Message 11	6233	141	232	186	1	2	1.2
Message 12	3117	10171	10246	10209	2	3	2.4
Message 13	6233	5203	5277	5240	1	2	1.2
Message 14	3117	216	290	253	2	3	2.4
Message 15	6233	230	321	275	1	2	1.2
Message 16	6233	5306	5381	5344	1	2	1.2
Message 17	6233	312	386	349	1	2	1.2
Message 18	6233	5341	5416	5379	1	2	1.2
Message 19	623	15346	15420	15384	11	13	13
Message 20	1246	5377	5451	5414	4	7	7
Message 21	624	10416	10489	10453	11	13	13
Message 22	624	429	503	466	11	13	13
Message 23	624	10477	10551	10514	11	13	13
Message 24	249	619	692	655	29	31	31
Message 25	125	650	722	686	59	62	61
Message 26	249	5563	5636	5599	29	31	31
Message 27	6233	567	675	624	1	2	1.2
Message 28	623	5598	5671	5635	11	13	13
Message 29	623	5629	5728	5687	11	13	13
Message 30	623	15685	15759	15722	11	13	13
Message 31	32	45719	45787	45753	247	248	248
Message 32	32	35723	35791	35758	247	248	248
Message 33	62	25725	25797	25760	122	124	123
Message 34	63	15809	15881	15849	122	124	123
Message 35	3117	933	1032	993	2	3	2.4
Message 36	32	65817	65888	65857	247	248	248
Message 37	32	45874	45946	45913	247	248	248
Message 38	31	910	980	946	247	248	248
Message 39	31	70957	71027	70991	247	248	248
Message 40	31	31116	31187	31152	247	248	248
Message 41	623	15997	16071	16035	11	13	13

Table A-3: Static Segment Experiment 3: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (µs)	Max. Delay (µs)	Av. Delay (µs)	Min. Jitter (µs)	Max. Jitter (µs)	Av. Jitter (µs)
Message 1	6044	5304	5378	5341	1	2	1.2
Message 2	12088	256	347	301	0	1	0.6

Table A-3 (continued)

Message 3	3022	410	484	447	2	3	2.4
Message 4	6044	5269	5360	5315	1	2	1.2
Message 5	6044	5056	5130	5093	1	2	1.2
Message 6	6044	5301	5375	5338	1	2	1.2
Message 7	6044	5054	5129	5092	1	2	1.2
Message 8	6044	5060	5134	5097	1	2	1.2
Message 9	6044	5283	5374	5328	1	2	1.2
Message 10	6044	5314	4388	5351	1	2	1.2
Message 11	6044	5296	5387	5341	1	2	1.2
Message 12	3022	10388	10463	10426	2	3	2.4
Message 13	6044	5327	5401	5364	1	2	1.2
Message 14	3022	10340	10414	10377	2	3	2.4
Message 15	6044	5292	5383	5337	1	2	1.2
Message 16	6044	5058	5133	5096	1	2	1.2
Message 17	6044	5064	5138	5100	1	2	1.2
Message 18	6044	5062	5137	5100	1	2	1.2
Message 19	605	114	172	143	11	13	13
Message 20	1209	66	140	103	4	7	7
Message 21	605	15354	15428	15391	11	13	13
Message 22	605	10336	10410	10373	11	13	13
Message 23	605	5353	5427	5391	11	13	13
Message 24	242	5525	5599	5562	29	31	31
Message 25	121	5556	5629	5592	59	62	61
Message 26	242	5067	5140	5104	29	31	31
Message 27	6044	5288	5396	5345	1	2	1.2
Message 28	605	15070	15144	15113	11	13	13
Message 29	605	82	169	134	11	13	13
Message 30	605	15096	15170	15135	11	13	13
Message 31	31	15171	15192	15182	239	242	242
Message 32	31	10140	10160	10151	239	242	242
Message 33	61	5125	5146	5136	120	122	122
Message 34	61	5158	5179	5168	120	122	122
Message 35	3022	10529	10629	10590	2	3	2.4
Message 36	30	60092	60112	60103	239	242	242
Message 37	30	45110	45131	45121	239	242	242
Message 38	30	40352	40422	40387	239	242	242
Message 39	30	40374	40433	40405	239	242	242
Message 40	30	65574	65628	65602	239	242	242
Message 41	605	114	172	142	11	13	13

Table A-4: Static Segment Experiment 4: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (µs)	Max. Delay (µs)	Av. Delay (µs)	Min. Jitter (µs)	Max. Jitter (µs)	Av. Jitter (µs)
Message 1	6224	242	316	280	1	2	1.2
Message 2	12448	194	284	239	0	1	0.6
Message 3	3112	10348	10422	10385	2	3	2.5
Message 4	6224	207	298	252	1	2	1.2
Message 5	6224	5056	5130	5093	1	2	1.2
Message 6	6224	238	312	276	1	2	1.2
Message 7	6224	5054	5129	5092	1	2	1.2

Table A-4 (continued)

Message 8	6224	5060	5134	5097	1	2	1.2
Message 9	6224	220	311	266	1	2	1.2
Message 10	6224	252	326	289	1	2	1.2
Message 11	6224	234	325	279	1	2	1.2
Message 12	3112	15326	15401	15364	2	3	2.5
Message 13	6224	265	339	302	1	2	1.2
Message 14	3112	278	352	315	2	3	2.5
Message 15	6224	230	321	275	1	2	1.2
Message 16	6224	5058	5133	5096	1	2	1.2
Message 17	6224	5064	5138	5100	1	2	1.2
Message 18	6224	5062	5137	5100	1	2	1.2
Message 19	622	91	60141	30169	20010	59980	29986
Message 20	1245	5074	35140	20092	10005	29990	15001
Message 21	622	292	60365	30328	20010	59980	30051
Message 22	622	15274	75347	45375	20010	59980	29986
Message 23	622	10291	70364	40392	20010	59980	29986
Message 24	249	5457	155452	80731	69959	90021	78777
Message 25	124	453	300505	151768	139919	180041	157533
Message 26	249	87	150106	74842	69959	90021	78777
Message 27	6224	5226	5334	5283	1	2	1.2
Message 28	622	5040	65108	35076	20010	59980	30051
Message 29	622	10060	70139	40162	20010	59980	29986
Message 30	622	5097	65139	35038	20010	59980	29986
Message 31	31	20067	260135	143978	80242	239718	117453
Message 32	31	15042	255110	138955	80241	239718	117453
Message 33	62	10016	290084	152626	40120	279839	71559
Message 34	62	10108	290133	153988	40120	279839	67630
Message 35	3112	5409	5505	5466	2	3	2.4
Message 36	32	65038	305079	185059	80242	239718	116253
Message 37	32	50056	290097	170077	80241	239718	116252
Message 38	31	30273	270319	146424	80241	239718	117453
Message 39	31	25289	265334	144019	80242	239718	122769
Message 40	31	40480	280495	159197	80242	239718	122769
Message 41	623	5036	65101	35121	20010	59980	29970

Table A-5: Delay values for the case: 3 Nodes, 15 Messages, 8 bytes of payload, 472 Minislots, 53 Messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	211	24	5005	2486
Message 66	206	36	5013	2408
Message 67	208	61	5018	2734
Message 68	217	130	4988	2495
Message 69	213	33	4952	2352
Message 70	213	49	4982	2435
Message 71	211	59	4928	2391
Message 72	213	14	4972	2610
Message 73	211	15	4996	2491
Message 74	208	25	5010	2554
Message 75	213	34	5028	2440
Message 76	207	28	5000	2439

Table A-5 (continued)

Message 77	210	46	5015	2457
Message 78	209	29	5018	2529
Message 79	215	49	5033	2560

Table A-6: Delay values for the case: 3 Nodes, 15 Messages, 64 bytes of payload, 472 Minislots, 53 Messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	212	34	4991	2446
Message 66	220	32	5018	2538
Message 67	221	69	4985	2533
Message 68	222	53	5016	2533
Message 69	215	34	4974	2578
Message 70	220	72	4986	2425
Message 71	211	29	4965	2537
Message 72	215	57	5011	2533
Message 73	225	58	5077	2661
Message 74	221	31	4995	2518
Message 75	214	24	5055	2608
Message 76	234	38	5020	2463
Message 77	214	100	5092	2392
Message 78	217	37	4996	2439
Message 79	218	27	5014	2479

Table A-7: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 472 Minislots, 53 Messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	408	86	5021	2573
Message 66	428	32	5019	2518
Message 67	395	27	5008	2451
Message 68	416	27	5281	2531
Message 69	408	23	5009	2566
Message 70	408	35	5261	2519
Message 71	427	41	5210	2502
Message 72	395	15	5198	2524
Message 73	416	27	5198	2488
Message 74	409	37	5244	2667
Message 75	409	24	5243	2459
Message 76	431	31	5529	2494
Message 77	394	88	5398	2546
Message 78	416	37	5491	2550
Message 79	410	26	5277	2640

Table A-8: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 141 Minislots, 53 Messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	402	27	5001	2490
Message 66	418	36	5007	2451
Message 67	385	24	5011	2553
Message 68	409	26	5025	2504
Message 69	403	35	5274	2457
Message 70	402	15	5012	2503
Message 71	420	15	8974	2613
Message 72	386	39	5251	2625
Message 73	408	25	10098	2473
Message 74	402	16	8483	2443
Message 75	402	22	8305	2513
Message 76	422	23	9448	2579
Message 77	387	24	9236	2703
Message 78	407	24	9666	2610
Message 79	402	25	12631	2590

Table A-9: Delay values for the case: 3 Nodes, 15 Messages, 254 bytes of payload, 141 Minislots, 300 Messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	2375	29	5022	2617
Message 66	2371	275	5268	2858
Message 67	2372	521	5515	3103
Message 68	2372	3991	9951	7533
Message 69	2370	4267	10197	7779
Message 70	2376	14	5014	2470
Message 71	2375	270	5269	2721
Message 72	2375	4985	9984	7436
Message 73	2373	4292	10240	7688
Message 74	2374	9955	24953	14379
Message 75	2376	23	5021	2556
Message 76	2376	269	9029	2803
Message 77	2376	4045	9972	7504
Message 78	2376	4322	14046	7752
Message 79	2375	9056	14923	12454

Table A-10: Static Segment Experiment 6: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)	Min. Jitter (μs)	Max. Jitter (μs)	Av. Jitter (μs)
Message 1	6258	304	378	341	1	2	1.2
Message 2	12516	256	346	301	0	1	0.6
Message 3	3129	5410	5484	5447	2	3	2.4

Table A-10 (continued)

Message 4	6258	269	360	314	1	2	1.2
Message 5	6258	5055	5129	5093	1	2	1.2
Message 6	6258	301	374	338	1	2	1.2
Message 7	6258	5054	5129	5092	1	2	1.2
Message 8	6258	5059	5133	5096	1	2	1.2
Message 9	6258	282	373	328	1	2	1.2
Message 10	6258	314	388	351	1	2	1.2
Message 11	6258	296	387	341	1	2	1.2
Message 12	3129	15389	15463	15426	2	3	2.4
Message 13	6258	327	401	364	1	2	1.2
Message 14	3129	15340	15414	15377	2	3	2.4
Message 15	6258	292	383	337	1	2	1.2
Message 16	6258	5058	5133	5096	1	2	1.2
Message 17	6258	5063	5137	5100	1	2	1.2
Message 18	6258	5062	5137	5100	1	2	1.2
Message 19	626	98	172	134	12	13	12
Message 20	1251	67	141	104	6	7	6
Message 21	625	354	427	391	12	13	12
Message 22	626	15336	15410	15373	12	13	12
Message 23	626	10353	10426	10390	12	13	12
Message 24	250	5526	5599	5562	30	31	31
Message 25	125	5556	5630	5592	61	62	62
Message 26	251	5066	5140	5103	30	31	31
Message 27	6258	288	396	345	1	2	1.2
Message 28	626	15070	15144	15107	12	13	12
Message 29	626	69	170	128	12	13	12
Message 30	626	15096	15170	15133	12	13	12
Message 31	32	15127	15199	15168	247	248	247
Message 32	32	10100	10172	10140	247	248	247
Message 33	63	5073	5147	5115	123	124	124
Message 34	63	5126	5198	5163	123	124	124
Message 35	3129	10529	10629	10590	2	3	2.4
Message 36	32	60072	60145	60105	247	248	247
Message 37	32	45099	45171	45129	247	248	247
Message 38	31	5353	5423	5387	247	248	247
Message 39	31	5373	5433	5403	247	248	247
Message 40	31	45556	45628	45593	247	248	247
Message 41	626	98	172	135	12	13	12
Message 42	3129	10585	10659	10622	2	3	2.4
Message 43	125	10585	10658	10622	61	62	62
Message 44	625	10585	10658	10621	12	13	12
Message 45	251	5615	5689	5652	30	31	31
Message 46	6258	588	689	650	1	2	1.2
Message 47	125	15652	15725	15689	61	62	62
Message 48	250	5656	5756	5706	30	31	31
Message 49	6258	5682	5756	5719	1	2	1.2
Message 50	125	5717	5789	5752	61	62	62
Message 51	125	5716	5789	5753	61	62	62

Table A-11: Static Segment Experiment 7: Delay and Jitter values

Message ID	Number of Messages	Min. Delay (µs)	Max. Delay (µs)	Av. Delay (µs)	Min. Jitter (µs)	Max. Jitter (µs)	Av. Jitter (µs)
Message 1	6537	242	316	279	1	2	1.2
Message 2	13073	194	284	239	0	1	0.6
Message 3	3268	10348	10422	10385	2	3	2.4
Message 4	6536	207	298	252	1	2	1.2
Message 5	6537	5055	5129	5093	1	2	1.2
Message 6	6537	239	312	276	1	2	1.2
Message 7	6536	5054	5129	5092	1	2	1.2
Message 8	6537	5059	5133	5096	1	2	1.2
Message 9	6536	220	311	266	1	2	1.2
Message 10	6537	252	326	289	1	2	1.2
Message 11	6536	234	325	279	1	2	1.2
Message 12	3268	15326	15401	15363	2	3	2.4
Message 13	6537	265	339	302	1	2	1.2
Message 14	3268	278	352	315	2	3	2.4
Message 15	6536	229	321	275	1	2	1.2
Message 16	6536	5058	5133	5096	1	2	1.2
Message 17	6537	5063	5137	5100	1	2	1.2
Message 18	6536	5062	5137	5100	1	2	1.2
Message 19	654	67	60140	30043	20009	59981	29987
Message 20	1307	5067	35141	20115	10004	29991	14993
Message 21	654	292	60364	30328	20009	59981	19987
Message 22	654	15275	75347	45250	20009	59981	29987
Message 23	654	10292	70363	40266	20009	59981	29987
Message 24	262	5433	155505	80240	69960	90020	78722
Message 25	130	447	300505	151084	139920	180040	157648
Message 26	261	72	150137	75428	69960	90020	78755
Message 27	6537	5226	5334	5283	1	2	1.2
Message 28	654	5040	65113	35076	20009	59981	29987
Message 29	654	10041	70140	40036	20009	59981	29987
Message 30	654	5066	65139	35162	20009	59981	29987
Message 31	32	20069	260137	140100	80238	239723	121395
Message 32	32	15040	255108	135074	80238	239723	121395
Message 33	65	10011	290075	151582	40119	279842	70084
Message 34	65	10071	290137	151026	40118	279842	70084
Message 35	3269	5405	5505	5466	2	3	2.4
Message 36	32	65019	305072	185053	80238	239723	121395
Message 37	32	50037	290106	170079	80238	239723	121395
Message 38	33	10284	250314	129082	80238	239723	121395
Message 39	33	5274	245344	121673	80238	239723	120109
Message 40	33	20439	260490	144103	80237	239723	120109
Message 41	653	5036	65110	35027	20009	59981	30002
Message 42	3268	461	535	498	2	3	2.4
Message 43	131	5463	305526	156796	139920	180040	157511
Message 44	653	461	60534	30544	20009	59981	30002
Message 45	261	5471	155525	80822	69960	90020	78755
Message 46	6537	433	534	495	1	2	1.2
Message 47	131	15500	315569	164694	139920	180040	157511
Message 48	261	5472	155564	80616	69960	90020	78755
Message 49	6537	5496	5570	5533	1	2	1.2
Message 50	130	10535	310599	160567	139920	180040	157648
Message 51	130	10532	310603	160567	139920	180040	157648

Table A-12: Delay Values for the case 6 Nodes, 16 Bytes of payload data, 472 minislots, 188 messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	427	56	5014	2551
Message 66	411	33	5009	2563
Message 67	424	49	5018	2521
Message 68	431	20	5015	2457
Message 69	415	21	5013	2584
Message 70	429	23	5018	2500
Message 71	442	31	5044	2431
Message 72	422	19	5007	2403
Message 73	418	21	4989	2553
Message 74	427	14	5030	2530
Message 75	425	29	5009	2498
Message 76	428	15	5039	2491
Message 77	430	24	5020	2492
Message 78	420	72	5020	2491
Message 79	424	24	5035	2687
Message 80	419	32	5036	2468
Message 81	429	23	5060	2530
Message 82	427	23	5049	2529
Message 83	442	26	4982	2518
Message 84	423	25	5041	2504
Message 85	417	29	5044	2411
Message 86	427	34	5037	2630
Message 87	425	25	5026	2478
Message 88	426	27	5064	2587
Message 89	441	41	5072	2413
Message 90	424	54	5090	2503
Message 91	418	23	4997	2467
Message 92	425	28	5018	2521
Message 93	423	40	5029	2544
Message 94	428	35	5088	2502
Message 95	429	40	5012	2483
Message 96	417	23	5040	2505
Message 97	421	24	5089	2549
Message 98	418	35	5032	2643
Message 99	430	23	4994	2580
Message 100	429	22	5064	2493

Table A-13: Delay Values for the case 6 Nodes, 36 Messages, 64 Bytes of payload data, 472 minislots, 188 messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	435	27	4990	2563
Message 66	432	26	5012	2650
Message 67	441	23	5012	2421
Message 68	412	43	5060	2505
Message 69	420	25	5003	2596
Message 70	1685	21	5103	2525
Message 71	422	19	5062	2429

Table A-13 (continued)

Message 72	427	25	5087	2514
Message 73	423	18	5000	2583
Message 74	424	15	5080	2548
Message 75	427	39	5011	2548
Message 76	1702	20	5166	2488
Message 77	422	25	5019	2612
Message 78	428	51	5063	2535
Message 79	425	56	5016	2577
Message 80	424	27	5159	2501
Message 81	429	37	5151	2499
Message 82	1701	21	5180	2498
Message 83	422	33	5111	2567
Message 84	426	25	5269	2517
Message 85	421	29	5153	2576
Message 86	426	23	5068	2517
Message 87	422	45	5067	2530
Message 88	1698	24	5255	2477
Message 89	421	67	5252	2490
Message 90	425	25	5135	2669
Message 91	424	31	5124	2507
Message 92	425	45	5100	2497
Message 93	427	30	5165	2597
Message 94	1699	24	5161	2539
Message 95	421	30	5053	2629
Message 96	426	22	5096	2427
Message 97	425	54	5167	2647
Message 98	423	54	5116	2432
Message 99	428	36	5110	2434
Message 100	1699	24	5255	2538

Table A-14: Delay Values for the case 6 Nodes, 36 Messages, 254 Bytes of payload data, 472 minislots, 188 messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	438	29	5002	2433
Message 66	430	49	5289	2424
Message 67	440	26	5015	2531
Message 68	415	32	5269	2486
Message 69	418	31	5228	2517
Message 70	1678	26	5287	2518
Message 71	420	21	5397	2587
Message 72	425	17	5251	2508
Message 73	424	17	5214	2595
Message 74	422	20	5175	2491
Message 75	428	22	5335	2473
Message 76	1702	15	5289	2512
Message 77	422	33	5559	2620
Message 78	426	24	5358	2473
Message 79	424	24	5183	2571
Message 80	424	31	5225	2412
Message 81	430	38	5240	2456

Table A-14 (continued)

Message 82	1700	22	5416	2558
Message 83	421	22	5597	2539
Message 84	426	24	5689	2504
Message 85	423	24	5471	2477
Message 86	425	26	5269	2575
Message 87	426	27	5465	2563
Message 88	1699	23	5556	2491
Message 89	420	23	5791	2612
Message 90	424	34	5499	2442
Message 91	425	25	5809	2612
Message 92	424	58	5557	2514
Message 93	427	24	5603	2389
Message 94	1697	27	5794	2559
Message 95	423	41	5759	2590
Message 96	433	31	5448	2605
Message 97	426	26	5553	2502
Message 98	423	29	5636	2522
Message 99	435	23	5500	2556
Message 100	1694	21	5829	2521

Table A-15: Delay Values for the case 6 Nodes, 36 Messages, 254 bytes payload, 141 minislots, 188 messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	409	40	5014	2560
Message 66	430	29	5232	2475
Message 67	424	24	5283	2533
Message 68	427	23	5161	2369
Message 69	422	24	5235	2445
Message 70	1699	22	5524	2550
Message 71	436	29	8690	2632
Message 72	420	21	7849	2598
Message 73	413	20	8689	2565
Message 74	433	21	10027	2594
Message 75	412	20	9156	2636
Message 76	1695	18	10160	2615
Message 77	431	28	14357	2719
Message 78	437	27	12400	2688
Message 79	419	71	9808	2726
Message 80	420	22	10046	2755
Message 81	440	45	9973	2856
Message 82	1723	21	16953	2874
Message 83	436	61	17464	2943
Message 84	421	22	9887	2797
Message 85	415	23	16006	3078
Message 86	434	36	13294	2922
Message 87	413	49	13704	3135
Message 88	1698	21	23229	3258
Message 89	438	23	19305	3640
Message 90	420	24	22459	3515
Message 91	414	50	17401	3685

Table A-15 (continued)

Message 92	434	26	16183	3387
Message 93	412	23	21092	3491
Message 94	1699	31	24456	3832
Message 95	421	40	24140	3958
Message 96	423	60	26903	4232
Message 97	430	37	20044	3954
Message 98	432	34	18151	4000
Message 99	422	29	23755	4319
Message 100	1691	29	45286	4520

Table A-16: Delay Values for the case 6 Nodes, 36 Messages, 254 bytes payload, 141 minislots 719 messages/sec

Message ID	Number of Messages	Min. Delay (μs)	Max. Delay (μs)	Av. Delay (μs)
Message 65	1484	34	5023	2513
Message 66	1480	23	5237	2540
Message 67	1478	30	5535	2547
Message 68	1487	22	6771	2480
Message 69	1484	22	9613	2475
Message 70	5941	22	10150	2534
Message 71	1478	16	13822	3165
Message 72	1486	23	17630	3167
Message 73	1481	17	20422	3509
Message 74	1475	16	19798	3648
Message 75	1488	22	20286	3968
Message 76	5885	15	21745	4166
Message 77	1497	23	32676	5879
Message 78	1478	40	37975	6733
Message 79	1479	31	39837	7363
Message 80	1484	22	42935	8307
Message 81	1488	34	48937	9435
Message 82	4686	26	24569	7009
Message 83	1489	54	91468	18374
Message 84	1429	21	92483	22276
Message 85	1317	22	96127	27872
Message 86	1041	66	98307	33012
Message 87	570	255	97420	36065
Message 88	207	42	74283	9242
Message 89	32	1397	97520	37821
Message 90	4	10792	69435	30782
Message 91	0	0	0	0
Message 92	0	0	0	0
Message 93	0	0	0	0
Message 94	0	0	0	0
Message 95	0	0	0	0
Message 96	0	0	0	0
Message 97	0	0	0	0
Message 98	0	0	0	0
Message 99	0	0	0	0
Message 100	0	0	0	0

Table A-17: Varying Dynamic Segment Length – 50 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	931	96	5024	2495	1458
Message 98	960	25	5020	2549	1496
Message 99	972	28	9987	2673	1824
Message 100	940	22	14375	2820	2091
Message 101	959	34	16259	2854	1948
Message 102	977	27	14839	3047	2042
Message 103	956	36	17475	3059	2487
Message 104	975	14	23104	3146	2751
Message 105	960	40	24382	3425	2925
Message 106	943	30	18228	3568	3084
Message 107	952	26	20443	3672	3289
Message 108	974	100	25857	3843	3795
Message 109	919	45	29285	3858	4014
Message 110	956	47	25500	4206	3984
Message 111	978	32	31524	4324	4504
Message 112	940	51	37472	4484	4628
Message 113	960	24	34734	4728	4745
Message 114	967	26	35702	5060	5441
Message 115	923	24	44277	5087	5809
Message 116	951	25	44706	5389	5672
Message 117	985	67	47052	5870	6247
Message 118	937	27	45715	6107	7159
Message 119	962	34	54733	6275	7176
Message 120	978	23	50398	6187	6437
Message 121	918	50	59699	6992	8432
Message 122	952	31	60073	7581	9522
Message 123	0	N.A.	N.A.	N.A.	N.A.
Message 124	0	N.A.	N.A.	N.A.	N.A.
Message 125	0	N.A.	N.A.	N.A.	N.A.
Message 126	0	N.A.	N.A.	N.A.	N.A.
Message 127	0	N.A.	N.A.	N.A.	N.A.
Message 128	0	N.A.	N.A.	N.A.	N.A.
Message 129	0	N.A.	N.A.	N.A.	N.A.
Message 130	0	N.A.	N.A.	N.A.	N.A.
Message 131	0	N.A.	N.A.	N.A.	N.A.
Message 132	0	N.A.	N.A.	N.A.	N.A.

Table A-18: Varying Dynamic Segment Length – 100 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	25	5022	2490	1470
Message 98	945	63	5021	2621	1450
Message 99	1005	64	5019	2447	1495
Message 100	950	25	5154	2534	1520
Message 101	971	45	5064	2504	1481
Message 102	990	57	5144	2598	1453
Message 103	917	25	5102	2522	1442
Message 104	946	20	5135	2399	1483
Message 105	1001	29	5136	2509	1434

Table A-18 (continued)

Message 106	952	23	5516	2477	1430
Message 107	970	18	5114	2483	1500
Message 108	991	36	5132	2438	1512
Message 109	912	42	5128	2523	1509
Message 110	949	45	7230	2562	1469
Message 111	1006	39	9743	2550	1501
Message 112	947	37	7078	2656	1919
Message 113	976	29	5151	2521	1471
Message 114	996	73	7436	2598	1482
Message 115	918	33	9802	2509	1541
Message 116	946	28	8430	2562	1512
Message 117	1012	62	9371	2598	1580
Message 118	943	26	9459	2564	1505
Message 119	983	37	14802	2524	1611
Message 120	996	35	8450	2586	1479
Message 121	921	80	8250	2614	1579
Message 122	952	67	9031	2546	1483
Message 123	1016	45	11510	2674	1596
Message 124	943	25	13372	2675	1660
Message 125	978	37	19741	3012	2248
Message 126	1002	45	18665	3056	2260
Message 127	920	35	19799	3058	2181
Message 128	947	32	19527	3123	2694
Message 129	1021	29	18160	3185	2540
Message 130	946	23	14893	3170	2483
Message 131	973	37	19047	3258	2640
Message 132	998	38	15809	3135	2499

Table A-19: Varying Dynamic Segment Length – 200 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	921	30	5023		1476
Message 98	953	26	5003	2490	1452
Message 99	1011	34	5147	2479	1497
Message 100	942	24	5002	2519	1474
Message 101	981	22	5149	2471	1509
Message 102	993	26	5145	2549	1474
Message 103	924	29	5129	2558	1473
Message 104	940	25	5100	2585	1468
Message 105	961	23	5161	2557	1485
Message 106	923	29	5126	2471	1459
Message 107	972	19	5115	2613	1439
Message 108	980	42	5237	2550	1481
Message 109	925	23	5119	2595	1429
Message 110	947	34	5084	2557	1863
Message 111	1014	69	5117	2542	1471
Message 112	942	23	5133	2488	1432
Message 113	987	38	5278	2573	1495
Message 114	993	35	5171	2497	1448
Message 115	923	57	5299	2490	1460
Message 116	941	23	5215	2545	1486

Table A-19 (continued)

Message 117	1012	42	5115	2479	1495
Message 118	948	55	5056	2515	1489
Message 119	984	33	5265	2477	1457
Message 120	1002	65	5297	2513	1480
Message 121	923	49	5231	2492	1472
Message 122	943	39	5167	2606	1568
Message 123	1008	22	5156	2499	1447
Message 124	950	24	5155	2507	1551
Message 125	979	71	5227	2548	1578
Message 126	1003	52	5413	2555	1474
Message 127	927	49	5154	2535	1433
Message 128	946	29	5354	2485	1465
Message 129	1012	45	5333	2561	1483
Message 130	950	48	5278	2544	1434
Message 131	982	27	5508	2559	1492
Message 132	1001	25	5282	2445	1472

Table A-20: Varying Dynamic Segment Length – 307 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	25	5021	2533	1524
Message 98	950	27	5015	2485	1487
Message 99	1004	33	5116	2521	1517
Message 100	949	37	5007	2604	1439
Message 101	973	50	5074	2508	1458
Message 102	991	54	5163	2493	1500
Message 103	923	17	5008	2521	1472
Message 104	959	67	5144	2522	1489
Message 105	1003	17	5119	2541	1507
Message 106	945	30	5105	2496	1448
Message 107	966	19	5144	2524	1447
Message 108	987	21	5181	2456	1512
Message 109	920	93	5163	2490	1497
Message 110	948	23	5145	2515	1496
Message 111	1011	55	5246	2574	1483
Message 112	949	42	5171	2577	1833
Message 113	977	24	5108	2573	1442
Message 114	996	26	5204	2546	1515
Message 115	919	63	5132	2528	1484
Message 116	951	25	5118	2497	1507
Message 117	1015	24	5192	2558	1412
Message 118	951	46	5329	2589	1476
Message 119	978	24	5247	2461	1447
Message 120	1003	43	5248	2436	1462
Message 121	919	25	5295	2514	1470
Message 122	950	22	5240	2532	1496
Message 123	1020	25	5401	2403	1427
Message 124	948	49	5156	2483	1466
Message 125	980	22	5299	2473	1472
Message 126	996	23	5401	2513	1518
Message 127	925	22	5140	2560	1476

Table A-20 (continued)

Message 128	942	31	5172	2531	1458
Message 129	1018	46	5163	2529	1520
Message 130	952	29	5270	2620	1493
Message 131	977	35	5247	2497	1483
Message 132	1008	28	5145	2569	1425

Table A-21: Varying Payload Length – 16 Byte

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	945	24	5014	2510	1472
Message 98	963	33	5020	2563	1494
Message 99	988	41	5018	2532	1556
Message 100	942	43	5017	2453	1452
Message 101	963	73	5011	2502	1482
Message 102	929	39	5053	2484	1470
Message 103	950	14	5011	2436	1484
Message 104	967	17	5012	2533	1488
Message 105	983	20	5011	2553	1449
Message 106	944	24	5006	2495	1459
Message 107	974	36	5006	2529	1460
Message 108	923	24	5009	2450	1505
Message 109	949	38	5013	2501	1455
Message 110	963	45	5022	2446	1478
Message 111	980	24	5019	2477	1457
Message 112	937	25	5030	2623	1924
Message 113	965	34	5116	2500	1843
Message 114	945	29	5026	2513	1458
Message 115	947	33	5048	2519	1427
Message 116	959	31	5051	2524	1456
Message 117	971	58	5030	2481	1521
Message 118	932	58	5116	2555	1543
Message 119	957	36	5056	2576	1611
Message 120	943	48	5019	2538	1480
Message 121	946	42	5053	2538	1462
Message 122	959	42	5051	2576	1497
Message 123	976	27	5049	2446	1431
Message 124	938	33	5059	2466	1558
Message 125	957	44	5044	2485	1445
Message 126	944	30	5022	2492	1503
Message 127	954	38	5041	2463	1464
Message 128	957	48	5051	2554	1505
Message 129	975	32	5047	2566	1432
Message 130	944	44	5074	2471	1451
Message 131	961	37	5069	2600	1474
Message 132	945	28	5039	2517	1527

Table A-22: Varying Payload Length – 64 Byte

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	927	27	5019	2474	1446
Message 98	959	46	5021	2460	1480
Message 99	1019	23	5085	2437	1467
Message 100	955	63	5011	2439	1423
Message 101	979	28	5050	2489	1471
Message 102	997	41	5100	2470	1502
Message 103	932	24	5027	2460	1510
Message 104	960	25	5007	2492	1475
Message 105	996	16	5046	2470	1439
Message 106	963	30	5079	2524	1427
Message 107	981	37	5084	2485	1441
Message 108	970	15	5018	2498	1466
Message 109	928	50	5087	2589	1484
Message 110	955	42	5099	2532	1440
Message 111	1019	51	5180	2602	1517
Message 112	959	48	5101	2609	1436
Message 113	987	37	5097	2521	1546
Message 114	1009	26	5100	2497	1430
Message 115	931	28	5159	2461	1481
Message 116	955	43	5120	2517	1577
Message 117	1023	32	5019	2589	1463
Message 118	956	44	5028	2506	1480
Message 119	988	31	5102	2521	1458
Message 120	1010	24	5156	2493	1500
Message 121	926	25	5175	2564	1454
Message 122	957	30	4981	2561	1594
Message 123	1027	23	5189	2512	1490
Message 124	956	45	5132	2496	1442
Message 125	987	25	5173	2499	1466
Message 126	1006	41	5324	2482	1526
Message 127	928	22	5088	2537	1478
Message 128	959	27	5072	2521	1449
Message 129	1027	62	5178	2526	1529
Message 130	957	50	5095	2600	1498
Message 131	991	32	5185	2501	1491
Message 132	1004	104	5141	2591	1478

Table A-23: Varying Payload Length – 128 Byte

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	25	5021	2533	1524
Message 98	950	27	5015	2485	1487
Message 99	1004	33	5116	2521	1517
Message 100	949	37	5007	2604	1439
Message 101	973	50	5074	2508	1458
Message 102	991	54	5163	2493	1500
Message 103	923	17	5008	2521	1472
Message 104	959	67	5144	2522	1489
Message 105	1003	17	5119	2541	1507

Table A-23 (continued)

Message 106	945	30	5105	2496	1448
Message 107	966	19	5144	2524	1447
Message 108	987	21	5181	2456	1512
Message 109	920	93	5163	2490	1497
Message 110	948	23	5145	2515	1496
Message 111	1011	55	5246	2574	1483
Message 112	949	42	5171	2577	1833
Message 113	977	24	5108	2573	1442
Message 114	996	26	5204	2546	1515
Message 115	919	63	5132	2528	1484
Message 116	951	25	5118	2497	1507
Message 117	1015	24	5192	2558	1412
Message 118	951	46	5329	2589	1476
Message 119	978	24	5247	2461	1447
Message 120	1003	43	5248	2436	1462
Message 121	919	25	5295	2514	1470
Message 122	950	22	5240	2532	1496
Message 123	1020	25	5401	2403	1427
Message 124	948	49	5156	2483	1466
Message 125	980	22	5299	2473	1472
Message 126	996	23	5401	2513	1518
Message 127	925	22	5140	2560	1476
Message 128	942	31	5172	2531	1458
Message 129	1018	46	5163	2529	1520
Message 130	952	29	5270	2620	1493
Message 131	977	35	5247	2497	1483
Message 132	1008	28	5145	2569	1425

Table A-24: Varying Payload Length – 254 Byte

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	921	37	5024	2508	1469
Message 98	948	32	5176	2510	1481
Message 99	1008	57	5064	2422	1453
Message 100	948	44	5020	2438	1479
Message 101	968	26	5283	2510	1465
Message 102	989	37	5281	2523	1502
Message 103	943	26	5255	2567	1477
Message 104	966	26	5462	2462	1448
Message 105	963	19	5248	2506	1416
Message 106	937	32	5266	2584	1451
Message 107	982	28	5418	2461	1453
Message 108	970	39	5248	2452	1533
Message 109	922	36	5430	2571	1506
Message 110	949	44	5291	2473	1460
Message 111	1017	28	5537	2636	1806
Message 112	946	29	5192	2529	1466
Message 113	980	29	5302	2542	1438
Message 114	998	34	5246	2500	1416
Message 115	922	23	5861	2573	1477
Message 116	947	60	5230	2584	1540

Table A-24 (continued)

Message 117	1017	23	5534	2468	1517
Message 118	949	27	5503	2525	1605
Message 119	977	51	5547	2525	1609
Message 120	997	29	5530	2570	1483
Message 121	925	27	5827	2607	1490
Message 122	945	35	5498	2497	1472
Message 123	1017	37	5440	2679	1494
Message 124	952	43	5515	2580	1656
Message 125	972	30	5964	2560	1456
Message 126	1005	47	5322	2516	1437
Message 127	923	31	5537	2534	1496
Message 128	943	47	5792	2530	1489
Message 129	1011	34	5556	2525	1517
Message 130	954	38	5689	2562	1469
Message 131	968	45	5773	2562	1473
Message 132	1001	26	5816	2505	1479

Table A-25: Varying Message Traffic Rate – 480 messages/second

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	3642	24	5024	2537	1450
Message 98	3652	30	5129	2492	1441
Message 99	3627	27	5155	2551	1469
Message 100	3618	24	5150	2519	1453
Message 101	3627	34	5252	2478	1471
Message 102	3635	21	5298	2515	1441
Message 103	3625	22	5263	2494	1475
Message 104	3636	19	5371	2492	1472
Message 105	3623	19	5412	2479	1462
Message 106	3624	16	5402	2544	1466
Message 107	3659	21	5248	2489	1439
Message 108	3641	14	5403	2523	1440
Message 109	3645	28	5343	2544	1465
Message 110	3637	23	5349	2539	1513
Message 111	3628	27	5358	2530	1453
Message 112	3611	26	5484	2522	1480
Message 113	3628	23	5465	2552	1585
Message 114	3648	24	5540	2519	1458
Message 115	3624	25	5633	2520	1461
Message 116	3624	24	5414	2524	1492
Message 117	3648	27	5465	2578	1506
Message 118	3637	49	5480	2543	1478
Message 119	3642	22	5560	2555	1486
Message 120	3623	23	5431	2516	1499
Message 121	3621	24	5663	2563	1473
Message 122	3642	24	5535	2526	1515
Message 123	3647	24	5433	2566	1498
Message 124	3636	28	5591	2534	1506
Message 125	3648	30	5521	2529	1487
Message 126	3604	22	5797	2550	1472
Message 127	3636	22	5737	2489	1463

Table A-25 (continued)

Message 128	3636	23	5799	2567	1687
Message 129	3630	25	5741	2518	1456
Message 130	3611	24	5716	2517	1693
Message 131	3641	22	5647	2521	1463
Message 132	3642	33	5791	2482	1479

Table A-26: Varying Message Traffic Rate – 238 messages/second

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	1806	35	5019	2519	1489
Message 98	1797	29	5048	2550	1435
Message 99	1838	42	5096	2588	1480
Message 100	1819	22	5118	2514	1457
Message 101	1820	27	5192	2545	1460
Message 102	1790	22	5303	2514	1497
Message 103	1831	23	5139	2488	1464
Message 104	1795	15	5213	2490	1467
Message 105	1814	14	5292	2476	1489
Message 106	1815	22	5409	2470	1464
Message 107	1825	27	5266	2538	1458
Message 108	1805	22	5442	2498	1472
Message 109	1818	36	5300	2490	1463
Message 110	1784	27	5278	2585	1665
Message 111	1795	33	5180	2583	1481
Message 112	1813	30	5327	2548	1486
Message 113	1853	23	5411	2562	1485
Message 114	1800	22	5281	2487	1460
Message 115	1815	21	5394	2535	1472
Message 116	1786	24	5297	2507	1538
Message 117	1800	38	5204	2593	1473
Message 118	1798	31	5399	2517	1475
Message 119	1849	35	5354	2528	1470
Message 120	1804	30	5451	2498	1478
Message 121	1822	27	5290	2534	1487
Message 122	1777	25	5363	2512	1601
Message 123	1792	24	5481	2518	1499
Message 124	1797	29	5729	2557	1448
Message 125	1853	22	5534	2457	1481
Message 126	1793	30	5425	2497	1499
Message 127	1816	33	5482	2512	1476
Message 128	1781	23	5521	2568	1487
Message 129	1794	26	5153	2606	1674
Message 130	1809	23	5642	2518	1489
Message 131	1853	23	5308	2590	1473
Message 132	1790	32	5583	2499	1466

Table A-27: Varying Message Traffic Rate – 127 messages/second

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	25	5021	2533	1524
Message 98	950	27	5015	2485	1487
Message 99	1004	33	5116	2521	1517
Message 100	949	37	5007	2604	1439
Message 101	973	50	5074	2508	1458
Message 102	991	54	5163	2493	1500
Message 103	923	17	5008	2521	1472
Message 104	959	67	5144	2522	1489
Message 105	1003	17	5119	2541	1507
Message 106	945	30	5105	2496	1448
Message 107	966	19	5144	2524	1447
Message 108	987	21	5181	2456	1512
Message 109	920	93	5163	2490	1497
Message 110	948	23	5145	2515	1496
Message 111	1011	55	5246	2574	1483
Message 112	949	42	5171	2577	1833
Message 113	977	24	5108	2573	1442
Message 114	996	26	5204	2546	1515
Message 115	919	63	5132	2528	1484
Message 116	951	25	5118	2497	1507
Message 117	1015	24	5192	2558	1412
Message 118	951	46	5329	2589	1476
Message 119	978	24	5247	2461	1447
Message 120	1003	43	5248	2436	1462
Message 121	919	25	5295	2514	1470
Message 122	950	22	5240	2532	1496
Message 123	1020	25	5401	2403	1427
Message 124	948	49	5156	2483	1466
Message 125	980	22	5299	2473	1472
Message 126	996	23	5401	2513	1518
Message 127	925	22	5140	2560	1476
Message 128	942	31	5172	2531	1458
Message 129	1018	46	5163	2529	1520
Message 130	952	29	5270	2620	1493
Message 131	977	35	5247	2497	1483
Message 132	1008	28	5145	2569	1425

Table A-28: Varying Number of Messages – 36 messages (6 per node)

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	25	5021	2533	1524
Message 98	950	27	5015	2485	1487
Message 99	1004	33	5116	2521	1517
Message 100	949	37	5007	2604	1439
Message 101	973	50	5074	2508	1458
Message 102	991	54	5163	2493	1500
Message 103	923	17	5008	2521	1472
Message 104	959	67	5144	2522	1489
Message 105	1003	17	5119	2541	1507

Table A-28 (continued)

Message 106	945	30	5105	2496	1448
Message 107	966	19	5144	2524	1447
Message 108	987	21	5181	2456	1512
Message 109	920	93	5163	2490	1497
Message 110	948	23	5145	2515	1496
Message 111	1011	55	5246	2574	1483
Message 112	949	42	5171	2577	1833
Message 113	977	24	5108	2573	1442
Message 114	996	26	5204	2546	1515
Message 115	919	63	5132	2528	1484
Message 116	951	25	5118	2497	1507
Message 117	1015	24	5192	2558	1412
Message 118	951	46	5329	2589	1476
Message 119	978	24	5247	2461	1447
Message 120	1003	43	5248	2436	1462
Message 121	919	25	5295	2514	1470
Message 122	950	22	5240	2532	1496
Message 123	1020	25	5401	2403	1427
Message 124	948	49	5156	2483	1466
Message 125	980	22	5299	2473	1472
Message 126	996	23	5401	2513	1518
Message 127	925	22	5140	2560	1476
Message 128	942	31	5172	2531	1458
Message 129	1018	46	5163	2529	1520
Message 130	952	29	5270	2620	1493
Message 131	977	35	5247	2497	1483
Message 132	1008	28	5145	2569	1425

Table A-29: Varying Number of Messages – 48 messages (8 per node)

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	956	51	5023	2527	1485
Message 98	967	30	5066	2423	1456
Message 99	985	23	5016	2514	1462
Message 100	921	24	5020	2522	1438
Message 101	956	31	5128	2460	1454
Message 102	949	61	5128	2484	1484
Message 103	2285	31	5156	2474	1444
Message 104	2243	26	5148	2519	1465
Message 105	962	35	5096	2588	1479
Message 106	967	26	5153	2430	1466
Message 107	956	33	5007	2498	1446
Message 108	954	16	5290	2494	1459
Message 109	938	25	5050	2468	1447
Message 110	958	44	5141	2524	1464
Message 111	2261	20	5398	2508	1413
Message 112	2295	17	5146	2479	1466
Message 113	934	43	5152	2487	1480
Message 114	974	97	5294	2508	1518
Message 115	938	28	5150	2539	1513
Message 116	940	40	5938	2509	1495

Table A-29 (continued)

Message 117	960	29	5207	2535	1820
Message 118	939	33	5281	2542	1431
Message 119	2233	36	5315	2529	1469
Message 120	2259	23	5287	2508	1459
Message 121	952	80	5168	2449	1455
Message 122	992	39	5387	2542	1512
Message 123	944	22	5194	2501	1425
Message 124	932	43	5337	2483	1473
Message 125	961	64	5273	2597	1485
Message 126	956	36	5417	2524	1487
Message 127	2287	34	5537	2474	1444
Message 128	2254	47	5437	2512	1496
Message 129	933	41	5185	2428	1444
Message 130	969	21	5265	2558	1499
Message 131	959	38	5331	2545	1616
Message 132	938	32	5238	2627	1595
Message 133	959	39	5589	2598	1592
Message 134	948	50	5403	2503	1450
Message 135	2248	22	5421	2547	1454
Message 136	2248	35	5443	2481	1492
Message 137	932	26	5533	2552	1503
Message 138	963	36	5263	2571	1523
Message 139	961	23	5168	2502	1463
Message 140	939	110	5225	2462	1459
Message 141	960	28	5307	2550	1453
Message 142	949	51	5197	2533	1487
Message 143	2246	24	5264	2509	1451
Message 144	2249	26	5711	2545	1496

Table A-30: Varying Number of Messages – 66 messages (11 per node)

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	956	29	5023	2587	1464
Message 98	940	70	5016	2642	1442
Message 99	952	41	5030	2482	1465
Message 100	958	29	5101	2509	1478
Message 101	955	27	5095	2442	1488
Message 102	957	25	5153	2556	1445
Message 103	2232	25	5165	2544	1456
Message 104	2287	27	5213	2522	1481
Message 105	2313	28	5273	2507	1488
Message 106	2261	22	5160	2544	1451
Message 107	2317	35	5210	2483	1465
Message 108	954	15	5231	2622	1522
Message 109	977	31	5278	2582	1483
Message 110	948	24	5279	2603	1525
Message 111	970	20	5120	2488	1511
Message 112	950	32	5251	2529	1433
Message 113	973	26	5136	2442	1468
Message 114	2275	15	5268	2485	1495
Message 115	2304	19	5222	2509	1457

Table A-30 (continued)

Message 116	2261	18	5432	2503	1465
Message 117	2272	17	5549	2518	1460
Message 118	2314	20	5300	2526	1446
Message 119	952	31	5303	2455	1477
Message 120	950	40	5268	2498	1500
Message 121	926	27	5283	2577	1488
Message 122	942	29	5428	2486	1434
Message 123	952	85	5222	2545	1701
Message 124	959	37	5352	2489	1477
Message 125	2276	23	5539	2574	1459
Message 126	2258	26	5408	2510	1469
Message 127	2289	40	5441	2412	1453
Message 128	2251	32	5383	2496	1443
Message 129	2285	28	5381	2482	1498
Message 130	963	33	5274	2575	1466
Message 131	975	28	5429	2532	1528
Message 132	920	33	5319	2669	1511
Message 133	958	46	5311	2550	1503
Message 134	943	42	5220	2459	1465
Message 135	972	30	5216	2600	1490
Message 136	2250	33	5508	2521	1458
Message 137	2299	38	5576	2518	1479
Message 138	2308	21	5399	2518	1484
Message 139	2272	25	5361	2542	1473
Message 140	2291	23	5283	2525	1475
Message 141	966	51	5443	2502	1485
Message 142	945	27	5400	2563	1547
Message 143	936	42	5591	2596	1480
Message 144	952	41	5552	2565	1422
Message 145	946	31	5239	2529	1527
Message 146	963	32	5737	2445	1454
Message 147	2247	21	5523	2508	1457
Message 148	2257	21	5483	2456	1458
Message 149	2309	25	5621	2499	1454
Message 150	2270	31	5578	2532	1491
Message 151	2277	28	5482	2503	1468
Message 152	941	50	5425	2581	1450
Message 153	956	27	5576	2544	1655
Message 154	933	46	5653	2528	1485
Message 155	957	23	5535	2458	1491
Message 156	934	86	5439	2584	1466
Message 157	979	22	5263	2464	1507
Message 158	2261	23	5733	2522	1462
Message 159	2246	25	5659	2538	1483
Message 160	2307	27	5689	2559	1483
Message 161	2301	24	5767	2519	1460
Message 162	2282	25	5596	2492	1522

Table A-31: Varying Number of Messages – 84 messages (14 per node)

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	957	29	5023	2512	1500
Message 98	957	26	5131	2494	1441
Message 99	955	109	5018	2488	1468
Message 100	978	46	5022	2402	1466
Message 101	967	31	5129	2559	1466
Message 102	972	21	5143	2428	1477
Message 103	2269	33	5255	2465	1457
Message 104	2298	22	5137	2468	1465
Message 105	2294	23	5253	2497	1433
Message 106	2285	21	5273	2537	1451
Message 107	2288	22	5282	2502	1454
Message 108	2311	22	5290	2539	1500
Message 109	2321	22	5277	2541	1463
Message 110	2324	24	5260	2515	1477
Message 111	969	14	5179	2504	1484
Message 112	959	37	5098	2474	1462
Message 113	967	57	5150	2510	1424
Message 114	960	37	5162	2583	1482
Message 115	936	31	5360	2652	1502
Message 116	950	25	5350	2499	1496
Message 117	2292	38	5201	2555	1427
Message 118	2292	25	5266	2491	1445
Message 119	2312	22	5295	2540	1455
Message 120	2287	20	5325	2489	1462
Message 121	2343	16	5316	2536	1466
Message 122	2321	27	5420	2530	1478
Message 123	2256	17	5586	2535	1485
Message 124	2278	31	5349	2511	1455
Message 125	968	43	5248	2539	1595
Message 126	920	46	5136	2477	1529
Message 127	978	26	5283	2504	1514
Message 128	962	22	5340	2476	1534
Message 129	952	24	5272	2609	1472
Message 130	956	42	5426	2532	1480
Message 131	2318	28	5450	2544	1470
Message 132	2256	29	5576	2507	1498
Message 133	2303	22	5442	2536	1476
Message 134	2273	30	5305	2512	1467
Message 135	2295	25	5728	2547	1479
Message 136	2311	31	5374	2512	1542
Message 137	2283	24	5642	2518	1472
Message 138	2260	22	5640	2560	1471
Message 139	989	23	5488	2551	1514
Message 140	966	25	5374	2522	1492
Message 141	940	25	5436	2461	1499
Message 142	933	40	5134	2561	1482
Message 143	945	42	5261	2560	1496
Message 144	931	22	5690	2560	1455
Message 145	2270	26	5618	2603	1488
Message 146	2230	25	5715	2547	1486
Message 147	2317	26	5663	2536	1455
Message 148	2270	28	5588	2557	1482

Table A-31 (continued)

Message 149	2276	29	5601	2533	1488
Message 150	2324	35	5435	2503	1477
Message 151	2317	27	5639	2515	1460
Message 152	2273	29	5631	2521	1497
Message 153	968	33	5436	2421	1472
Message 154	967	22	5430	2523	1477
Message 155	947	68	5670	2616	1414
Message 156	939	24	5707	2613	1526
Message 157	946	27	5430	2559	1434
Message 158	938	37	5586	2492	1458
Message 159	2303	45	5578	2532	1474
Message 160	2209	22	5703	2526	1490
Message 161	2313	49	5833	2550	1483
Message 162	2273	28	5567	2444	1456
Message 163	2272	26	5672	2588	1452
Message 164	2316	24	5551	2539	1487
Message 165	2283	23	5553	2494	1454
Message 166	2322	30	5926	2562	1486
Message 167	977	27	5534	2511	1451
Message 168	951	47	5497	2557	1505
Message 169	958	31	5584	2501	1569
Message 170	962	24	5683	2456	1505
Message 171	946	32	5394	2508	1516
Message 172	945	23	5587	2576	1524
Message 173	2328	22	5378	2486	1491
Message 174	2253	26	5877	2548	1506
Message 175	2290	24	5553	2499	1488
Message 176	2279	25	9506	2557	1520
Message 177	2294	23	8898	2540	1484
Message 178	2292	22	7591	2545	1693
Message 179	2292	23	9053	2536	1493
Message 180	2279	29	8559	2543	1482

Table A-32: Decreasing Payload Length – dynamic segment length of 100 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	928	31	5022	2489	1480
Message 98	970	25	5021	2500	1428
Message 99	992	23	5018	2493	1476
Message 100	939	27	8466	2534	1517
Message 101	973	24	7250	2510	1488
Message 102	996	51	7255	2567	1441
Message 103	942	23	9560	2527	1547
Message 104	953	24	8727	2587	1512
Message 105	1009	17	9225	2516	1599
Message 106	968	15	12843	2588	1599
Message 107	984	15	15161	3174	2688
Message 108	973	29	18974	3206	2664
Message 109	935	24	14101	2598	1708
Message 110	961	47	9079	2590	1622
Message 111	992	62	11509	2599	1633

Table A-32 (continued)

Message 112	945	27	13631	2644	1756
Message 113	973	26	14537	2679	2057
Message 114	983	58	10142	2599	1709
Message 115	935	36	9395	2693	1835
Message 116	960	48	10130	2640	1756
Message 117	991	59	13005	2757	1879
Message 118	950	49	9547	2784	1732
Message 119	970	36	13348	2699	1960
Message 120	978	23	14024	2713	1746
Message 121	932	49	9917	2729	1851
Message 122	959	28	19587	2720	1929
Message 123	990	33	11955	2646	1768
Message 124	950	37	12914	2741	1953
Message 125	975	23	14701	2692	1834
Message 126	972	66	19567	2912	2192
Message 127	928	27	9902	2696	1733
Message 128	957	28	10056	2734	1749
Message 129	986	22	10850	2638	1658
Message 130	945	45	13010	2717	1847
Message 131	975	47	14973	2669	1853
Message 132	967	33	13115	2743	2026

Table A-33: Decreasing Payload Length – dynamic segment length of 200 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	925	42	4994	2506	1480
Message 98	951	30	5021	2505	1453
Message 99	1003	42	5279	2419	1482
Message 100	948	48	5237	2539	1491
Message 101	988	28	5242	2406	1472
Message 102	992	31	5237	2674	1477
Message 103	925	37	5155	2454	1490
Message 104	954	21	5278	2578	1460
Message 105	1007	15	5376	2566	1496
Message 106	948	33	5352	2531	1499
Message 107	992	21	5398	2568	1517
Message 108	996	53	5279	2541	1430
Message 109	923	23	5290	2652	1462
Message 110	956	38	5300	2583	1449
Message 111	1013	26	5337	2521	1519
Message 112	945	45	5442	2568	1531
Message 113	992	34	5487	2500	1501
Message 114	995	43	5621	2509	1451
Message 115	924	25	5327	2521	1478
Message 116	953	39	5155	2544	1527
Message 117	1015	25	5339	2489	1487
Message 118	943	22	5098	2453	1500
Message 119	990	47	5480	2434	1447
Message 120	999	23	5436	2517	1536
Message 121	929	85	5727	2538	1454
Message 122	953	29	5494	2542	1484

Table A-33 (continued)

Message 123	1019	25	5308	2495	1442
Message 124	948	34	5416	2529	1477
Message 125	990	50	5398	2437	1434
Message 126	1002	22	5164	2500	1507
Message 127	930	76	5263	2548	1487
Message 128	953	37	5187	2547	1422
Message 129	1023	30	5361	2510	1504
Message 130	945	71	5287	2594	1496
Message 131	986	34	5468	2502	1472
Message 132	997	28	5508	2593	1440

Table A-34: Decreasing Payload Length – dynamic segment length of 307 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	43	5002	2566	1467
Message 98	954	29	5017	2438	1457
Message 99	1012	42	5280	2527	1502
Message 100	942	48	5113	2556	1436
Message 101	983	29	5274	2509	1502
Message 102	994	32	5237	2477	1468
Message 103	919	21	5314	2501	1461
Message 104	954	24	5275	2540	1446
Message 105	1012	20	5277	2540	1507
Message 106	941	20	5518	2529	1491
Message 107	980	16	5716	2565	1530
Message 108	991	31	5441	2473	1440
Message 109	922	22	5288	2457	1484
Message 110	953	64	5261	2495	1496
Message 111	1011	22	5421	2526	1479
Message 112	938	24	5278	2588	1844
Message 113	990	23	5247	2577	1488
Message 114	996	32	5285	2455	1517
Message 115	922	26	5350	2478	1494
Message 116	950	35	5538	2600	1494
Message 117	1008	25	5345	2442	1499
Message 118	943	30	5459	2485	1512
Message 119	988	48	5175	2493	1454
Message 120	994	36	5449	2584	1520
Message 121	926	50	5574	2505	1480
Message 122	948	97	5357	2573	1448
Message 123	1011	53	5319	2508	1601
Message 124	943	29	5153	2491	1606
Message 125	986	72	5051	2538	1597
Message 126	999	28	5290	2617	1473
Message 127	925	45	5383	2517	1483
Message 128	946	29	5459	2483	1495
Message 129	1013	78	5513	2637	1474
Message 130	946	47	5359	2538	1462
Message 131	982	28	5464	2539	1427
Message 132	998	29	5511	2527	1524

Table A-35: Increasing Payload Length – dynamic segment length of 100 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	28	5020	2506	1485
Message 98	953	24	5011	2492	1505
Message 99	1005	44	5019	2526	1485
Message 100	947	33	5003	2443	1489
Message 101	975	22	5013	2492	1492
Message 102	988	30	5016	2618	1443
Message 103	929	26	5008	2578	1532
Message 104	945	15	5031	2516	1459
Message 105	992	29	5046	2581	1472
Message 106	932	19	5007	2516	1444
Message 107	969	21	5021	2541	1434
Message 108	966	16	5022	2405	1471
Message 109	918	28	5055	2472	1471
Message 110	949	30	5066	2468	1487
Message 111	1007	56	5087	2488	1425
Message 112	938	56	5056	2508	1451
Message 113	983	26	5101	2481	1462
Message 114	992	49	5042	2464	1527
Message 115	920	27	5170	2535	1445
Message 116	948	23	8018	2494	1510
Message 117	1008	24	8066	2500	1455
Message 118	938	30	7528	2515	1563
Message 119	987	29	8390	2528	1493
Message 120	988	65	5099	2450	1510
Message 121	922	43	9911	2666	1678
Message 122	951	29	13203	2722	1892
Message 123	1012	25	18057	2879	1997
Message 124	941	32	19257	2855	2205
Message 125	986	28	18130	2984	2333
Message 126	996	39	14471	2911	2169
Message 127	922	51	13640	3128	2463
Message 128	943	79	19165	3953	3457
Message 129	1010	111	23379	4232	3811
Message 130	941	67	29530	4282	4318
Message 131	984	40	30809	4247	4118
Message 132	995	37	36256	4761	4533

Table A-36: Increasing Payload Length – dynamic segment length of 200 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	917	31	4994	2443	1484
Message 98	948	25	5017	2552	1448
Message 99	1000	30	5018	2507	1481
Message 100	947	36	5019	2561	1517
Message 101	975	33	5016	2496	1474
Message 102	985	67	5015	2576	1428
Message 103	916	30	5012	2530	1447
Message 104	942	32	5009	2453	1458
Message 105	988	16	5012	2575	1469

Table A-36 (continued)

Message 106	957	21	5049	2467	1451
Message 107	962	14	5042	2580	1431
Message 108	973	24	5024	2552	1427
Message 109	913	55	5043	2622	1473
Message 110	952	23	5033	2482	1442
Message 111	1003	31	5085	2590	1498
Message 112	936	32	5028	2604	1810
Message 113	979	29	5040	2547	1837
Message 114	991	23	5025	2499	1416
Message 115	917	38	5019	2496	1444
Message 116	941	31	5076	2577	1507
Message 117	1008	32	5015	2524	1543
Message 118	936	22	5068	2460	1571
Message 119	982	30	5159	2539	1504
Message 120	991	21	5182	2508	1502
Message 121	917	54	5159	2523	1482
Message 122	937	47	5091	2519	1470
Message 123	999	28	5103	2568	1589
Message 124	937	40	5167	2524	1474
Message 125	983	39	5227	2579	1706
Message 126	988	30	5354	2552	1498
Message 127	922	53	5455	2453	1498
Message 128	943	76	5295	2503	1507
Message 129	1008	33	5327	2568	1486
Message 130	939	29	5624	2514	1468
Message 131	980	80	6044	2576	1518
Message 132	991	28	5279	2479	1505

Table A-37: Increasing Payload Length – dynamic segment length of 307 minislots

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Std.Deviation
Message 97	918	38	5023	2517	1425
Message 98	949	58	5016	2466	1467
Message 99	1014	22	5013	2532	1436
Message 100	936	26	5020	2511	1421
Message 101	981	24	5009	2519	1467
Message 102	993	36	5020	2483	1490
Message 103	957	56	4993	2469	1443
Message 104	939	39	5041	2465	1529
Message 105	964	23	5026	2472	1452
Message 106	953	19	5036	2451	1509
Message 107	961	16	5045	2527	1425
Message 108	967	32	5042	2483	1508
Message 109	925	35	5139	2537	1490
Message 110	945	22	5046	2479	1483
Message 111	1015	37	5072	2576	1453
Message 112	941	32	5025	2553	1456
Message 113	982	40	5137	2515	1421
Message 114	987	39	5100	2537	1523
Message 115	922	22	5114	2461	1455
Message 116	944	21	5231	2551	1502

Table A-37 (continued)

Message 117	1008	49	5082	2498	1464
Message 118	942	47	5100	2477	1548
Message 119	982	42	5019	2499	1473
Message 120	999	45	5136	2554	1519
Message 121	923	25	5042	2488	1451
Message 122	934	26	5233	2568	1473
Message 123	1003	22	5186	2500	1472
Message 124	948	49	5126	2506	1480
Message 125	980	24	5119	2514	1572
Message 126	998	22	5283	2513	1506
Message 127	920	25	5268	2555	1440
Message 128	935	25	5162	2470	1467
Message 129	1000	42	5442	2596	1475
Message 130	945	39	5245	2507	1444
Message 131	979	34	5433	2581	1504
Message 132	992	27	5187	2469	1478

Table A-38: Static Segment Experiment 5, 3 Nodes, Signal Packing

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Min. Jitter	Max. Jitter	Av. Jitter
Message 1	10363	372	447	409	1	2	1
Message 2	20725	304	402	353	0	1	0
Message 3	5181	370	444	406	2	3	2
Message 4	10363	372	447	409	1	2	1
Message 5	10363	174	250	213	1	2	1
Message 6	10363	372	447	409	1	2	1
Message 7	10363	174	250	213	1	2	1
Message 8	10363	174	250	213	1	2	1
Message 9	10362	349	424	386	1	2	1
Message 10	10362	349	424	386	1	2	1
Message 11	10362	349	424	386	1	2	1
Message 12	5181	370	444	406	2	3	2
Message 13	10363	370	444	407	1	2	1
Message 14	5181	370	444	406	2	3	2
Message 15	10363	370	444	407	1	2	1
Message 16	10362	175	250	212	1	2	1
Message 17	10362	175	250	212	1	2	1
Message 18	10362	175	250	212	1	2	1
Message 19	1037	10183	10258	10219	12	13	12
Message 20	2073	178	254	216	6	7	6
Message 21	1036	369	443	405	12	13	12
Message 22	1036	369	443	405	12	13	12
Message 23	1036	369	443	405	12	13	12
Message 24	415	502	577	539	31	32	31
Message 25	207	10548	10622	10584	63	64	63
Message 26	415	138	252	195	31	32	31
Message 27	10363	370	444	407	1	2	1
Message 28	1037	10183	10258	10219	12	13	12
Message 29	1037	10183	10258	10219	12	13	12
Message 30	1036	10223	10297	10259	12	13	12
Message 31	51	50222	50295	50254	255	256	255
Message 32	51	50222	50295	50254	255	256	255
Message 33	103	10222	10296	10255	127	128	127

Table A-38 (continued)

Message 34	103	10222	10296	10255	127	128	127
Message 35	5182	10507	10622	10583	2	3	2
Message 36	51	50222	50295	50254	255	256	255
Message 37	51	50222	50295	50255	255	256	255
Message 38	52	391	461	424	255	256	255
Message 39	52	391	461	424	255	256	255
Message 40	52	30547	30619	30583	255	256	255
Message 41	1036	10223	10297	10259	12	13	12

Table A-39: Static Segment Experiment 8, 6 Nodes, 2.5ms Cycle Duration, Scheduling Offsets are synchronized with generation offsets

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Min. Jitter	Max. Jitter	Av. Jitter
Message 1	10131	122	172	146	1	2	1
Message 2	20261	122	172	146	0	1	0
Message 3	5066	213	265	239	2	3	2
Message 4	10131	103	171	137	1	2	1
Message 5	10131	124	172	147	1	2	1
Message 6	10131	166	216	191	1	2	1
Message 7	10130	126	170	148	1	2	1
Message 8	10131	124	172	147	1	2	1
Message 9	10130	166	216	190	1	2	1
Message 10	10131	147	215	182	1	2	1
Message 11	10131	165	215	190	1	2	1
Message 12	5066	179	229	203	2	3	2
Message 13	10131	178	228	203	1	2	1
Message 14	5065	177	228	203	2	3	2
Message 15	10130	177	228	202	1	2	1
Message 16	10131	125	170	147	1	2	1
Message 17	10131	189	238	213	1	2	1
Message 18	10130	191	235	213	1	2	1
Message 19	1013	235	256	249	12	13	12
Message 20	2027	196	237	216	6	7	6
Message 21	1013	178	227	202	12	13	12
Message 22	1013	222	272	247	12	13	12
Message 23	1014	221	272	246	12	13	12
Message 24	405	342	391	366	31	32	31
Message 25	202	344	392	367	63	64	63
Message 26	405	192	234	213	31	32	31
Message 27	10131	221	272	246	1	2	1
Message 28	1013	233	263	249	12	13	12
Message 29	1013	231	271	250	12	13	12
Message 30	1013	227	296	270	12	13	12
Message 31	50	224	265	244	254	255	254
Message 32	50	249	289	269	254	255	254
Message 33	101	256	287	269	127	128	127
Message 34	101	227	268	247	127	128	127
Message 35	5066	342	392	367	2	3	2
Message 36	50	252	292	271	254	255	254
Message 37	50	249	289	268	254	255	254
Message 38	51	221	270	245	254	255	254
Message 39	51	221	270	245	254	255	254
Message 40	51	348	390	367	254	255	254

Table A-39 (continued)

Message 41	1013	222	264	243	12	13	12
Message 42	5066	372	422	397	2	3	2
Message 43	203	372	422	396	63	64	63
Message 44	1013	372	421	396	12	13	12
Message 45	406	370	420	394	31	32	31
Message 46	10130	371	421	396	1	2	1
Message 47	203	408	457	432	63	64	63
Message 48	405	406	456	430	31	32	31
Message 49	10130	407	457	432	1	2	1
Message 50	203	440	489	464	63	64	63
Message 51	203	443	491	465	63	64	63

Table A-40: Static Segment Experiment 9, 6 Nodes, 2.5ms Cycle Duration, First occurrence of the messages are at the same cycle

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Min. Jitter	Max. Jitter	Av. Jitter
Message 1	10051	7618	7671	7645	1	2	1
Message 2	20102	101	173	138	0	1	0
Message 3	5026	12676	12729	12703	2	3	2
Message 4	10051	2566	2636	2602	1	2	1
Message 5	10051	7625	7669	7646	1	2	1
Message 6	10051	7611	7681	7647	1	2	1
Message 7	10051	98	142	119	1	2	1
Message 8	10051	2571	2615	2592	1	2	1
Message 9	10051	94	164	130	1	2	1
Message 10	10051	2577	2647	2613	1	2	1
Message 11	10051	5060	5130	5096	1	2	1
Message 12	5026	15074	15126	15101	2	3	2
Message 13	10051	7557	7643	7601	1	2	1
Message 14	5025	2539	2592	2566	2	3	2
Message 15	8135	43	126	84	1	2	1
Message 16	10051	5044	5089	5066	1	2	1
Message 17	10051	7611	7655	7632	1	2	1
Message 18	10051	84	128	105	1	2	1
Message 19	1005	17626	17650	17639	12	13	12
Message 20	2010	2537	2589	2564	6	7	6
Message 21	1005	5005	5057	5031	12	13	12
Message 22	1005	10050	10101	10076	12	13	12
Message 23	1005	12533	12584	12559	12	13	12
Message 24	402	7841	7891	7866	31	32	31
Message 25	201	314	364	339	63	64	63
Message 26	402	5011	5060	5035	31	32	31
Message 27	10051	7516	7670	7613	1	2	1
Message 28	1005	56	96	83	12	13	12
Message 29	1005	2519	2569	2553	12	13	12
Message 30	1005	4992	5042	5026	12	13	12
Message 31	50	29966	29986	29976	254	255	254
Message 32	50	32439	32459	32449	254	255	254
Message 33	101	34912	34986	34949	127	128	127
Message 34	101	9885	9959	9922	127	128	127
Message 35	5026	2788	2892	2865	2	3	2
Message 36	50	52359	52379	52369	254	255	254
Message 37	50	54832	54852	54842	254	255	254

Table A-40 (continued)

Message 38	51	74999	75046	75023	254	255	254
Message 39	51	79981	80029	80006	254	255	254
Message 40	50	5261	5311	5285	254	255	254
Message 41	1005	7305	7514	7487	12	13	12
Message 42	5026	5369	5422	5396	2	3	2
Message 43	201	7843	7893	7867	63	64	63
Message 44	1005	15316	15395	15363	12	13	12
Message 45	402	289	419	354	31	32	31
Message 46	10051	2762	2921	2878	1	2	1
Message 47	201	17906	17955	17930	63	64	63
Message 48	402	2879	2955	2917	31	32	31
Message 49	10051	5352	5457	5430	1	2	1
Message 50	201	10439	10489	10464	63	64	63
Message 51	201	12913	12962	12937	63	64	63

Table A-41: Static Segment Experiment 10, 6 Nodes, Signal Packing

Message ID	Number of Messages	Min. Delay	Max. Delay	Av. Delay	Min. Jitter	Max. Jitter	Av. Jitter
Message 1	9266	275	350	312	1	2	1
Message 2	18532	230	321	276	0	1	0
Message 3	4633	10376	10450	10413	2	3	2
Message 4	9266	244	334	289	1	2	1
Message 5	9264	46	120	83	1	2	1
Message 6	9266	272	347	310	1	2	1
Message 7	9265	46	120	83	1	2	1
Message 8	9266	50	125	88	1	2	1
Message 9	9266	257	347	302	1	2	1
Message 10	9266	285	360	323	1	2	1
Message 11	9266	270	360	315	1	2	1
Message 12	4633	10328	10402	10365	2	3	2
Message 13	9266	298	373	336	1	2	1
Message 14	4633	10340	10415	10378	2	3	2
Message 15	9266	283	357	320	1	2	1
Message 16	9266	50	125	87	1	2	1
Message 17	9266	54	130	92	1	2	1
Message 18	9266	55	130	92	1	2	1
Message 19	926	10087	10163	10125	12	13	12
Message 20	1853	45	126	81	6	7	6
Message 21	927	10326	10400	10362	12	13	12
Message 22	927	10340	10414	10377	12	13	12
Message 23	927	10325	10399	10361	12	13	12
Message 24	370	5482	5556	5519	31	32	31
Message 25	185	15512	15586	15549	62	63	62
Message 26	370	59	134	96	31	32	31
Message 27	9266	294	385	338	1	2	1
Message 28	926	10065	10162	10120	12	13	12
Message 29	926	10087	10161	10123	12	13	12
Message 30	926	10087	10161	10123	12	13	12
Message 31	47	50069	50143	50106	251	252	251
Message 32	47	50093	50163	50128	251	252	251
Message 33	93	10066	10140	10103	125	126	125
Message 34	93	10066	10140	10103	125	126	125
Message 35	4633	15487	15585	15547	2	3	2

Table A-41 (continued)

Message 36	47	50118	50188	50151	251	252	251
Message 37	47	50115	50189	50150	251	252	251
Message 38	46	10341	10401	10373	251	252	251
Message 39	46	10340	10410	10375	251	252	251
Message 40	46	75521	75576	75549	251	252	251
Message 41	926	10064	10159	10120	12	13	12
Message 42	4633	5567	5642	5604	2	3	2
Message 43	185	5568	5641	5604	62	63	62
Message 44	698	5567	5642	5603	12	13	12
Message 45	371	5515	5612	5563	31	32	31
Message 46	9266	5513	5612	5562	1	2	1
Message 47	185	629	703	665	62	63	62
Message 48	371	602	675	638	31	32	31
Message 49	9266	576	675	637	1	2	1
Message 50	185	10662	10736	10699	62	63	62
Message 51	185	10662	10736	10699	62	63	62