TOOL SUPPORT FOR WORST CASE END TO END DELAY ANALYSIS OF AFDX NETWORKS

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

BY

ORHUN EFE

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

FEBRUARY 2016

Approval of the thesis:

TOOL SUPPORT FOR WORST CASE END TO END DELAY ANALYSIS OF AFDX NETWORKS

submitted by **ORHUN EFE** 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. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı
Supervisor, **Electrical and Electronics Eng. Dept., METU** _____

Examining Committee Members:

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

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

Prof. Dr. Buyurman Baykal
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering Dept., METU _____

Prof. Dr. Bülent Tavlı
Electrical and Electronics Engineering Dept., TOBB-UET _____

Date: **05/02/2016**

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 : Orhun EFE

Signature :

ABSTRACT

TOOL SUPPORT FOR WORST CASE END TO END DELAY ANALYSIS OF AFDX NETWORKS

Efe, Orhun

M.S., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı

February 2016, 59 pages

Avionics Full Duplex Switched Ethernet (AFDX) is among the major technological components used in avionics systems. Since its publication, AFDX has spread out rapidly and has been deployed in major aircrafts such as Airbus A400M, Boeing 787, Bombardier C Series, etc. In AFDX networks, data is exchanged between end systems by utilizing tunnels. For certification purposes, finding a safe upper bound is required in transmission process. UPPAAL is already shown to be useful for performing such a delay analysis but it is tedious to create the state machines required. In the present thesis work, a software tool is developed to generate the state machines of a given AFDX network to enable users to find out the worst case end-to-end delay of a specified virtual link through using the analysis tool UPPAAL. In addition, the performance comparison of methods that are utilized for calculation of worst case end-to-end delay is done over specified network topologies. Furthermore, AFDX switch service disciplines, namely first-in-first-out (FIFO) and round-robin (RR), are compared in the above setting. This thesis work and the tool developed eliminate the need for finding out all of the candidate scenarios that are necessary to calculate the worst case end-to-end (E2E) delay. The delay value of an AFDX network can now be calculated with less user effort and time.

Keywords: AFDX network, Timed automata, UPPAAL modeling, Round robin scheduling, First-in-first-out scheduling, Worst-case E2E delay

ÖZ

AFDX AĞLARINDA EN BÜYÜK UÇTAN UCA GECİKME ÇÖZÜMLEMESİ İÇİN YAZILIM ARACI DESTEĞİ

Efe, Orhun

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Cüneyt F. Bazlamaçcı

Şubat 2016, 59 sayfa

Aviyonik Çift Yönlü Anahtarlamalı Ethernet (AFDX) aviyonik sistemlerde kullanılan ana teknolojik bileşenler arasındadır. Yayımlanmasından bu yana, AFDX hızla yaygınlaşmış ve başlıca Airbus A400M, Boeing 787, Bombardier C Serisi gibi başlıca hava araçlarında konuşlandırılmıştır. AFDX ağlarında, uç sistemler arası veri alış verişi tüneller aracılığı ile yapılmaktadır. Sertifikasyon amaçları kapsamında, uçtan uca gecikme zamanı için güvenli bir üst sınır bulunması zorunludur. UPPAAL'ın bu tip gecikme çözümlemesi yapabilmek için yararlı olduğu gösterilmiştir ancak gerekli durum makinalarını üretmek bıktırıcıdır. Bu tez çalışmasında, verilen bir AFDX ağının belirli bir sanal bağlantısına ait en büyük uçtan uca gecikme zamanını UPPAAL çözümleme aracını kullanarak hesaplamak için gereken ve AFDX ağının durum makinalarını oluşturacak kullanıcıya sunan bir yazılım aracı inşa edilmiştir. Buna ek olarak, belirli ağlarda en büyük uçtan uca gecikme zamanını hesaplamada kullanılan methodların performansları karşılaştırılmıştır. Dahası, AFDX anahtarlarda kullanılabilen iki servis disiplini, ilk-giren-ilk-çıkar (FIFO) ve dairesel denetim (RR) çizelgeleme algoritmaları için en büyük uçtan uca gecikme zamanı hesaplanmış ve karşılaştırılmıştır. Bu tez çalışması ve geliştirilmiş olan araç, en büyük uçtan uca gecikme zamanını hesaplamak için gerekli tüm aday olabilecek senaryoların ortaya çıkarılması gereksinimini ortadan

kaldırmıştır. Bir AFDX ağının en büyük uçtan uca gecikme zamanı artık daha az kullanıcı çabası ve zamanı ile hesaplanabilmektedir.

Anahtar Kelimeler: AFDX ağı, Zamanlı automata, UPPAAL modellemesi, Dairesel denetim çizelgelemesi, İlk-giren–ilk-çıkar çizelgelemesi, En büyük uçtan uca gecikme

To my beloved and my family

ACKNOWLEDGEMENTS

I express my sincere gratitude to Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı for his valuable supervision and support to my thesis work.

I would also like to thank to my company, ASELSAN for supporting me.

I also thank Turkish Scientific and Technological Research Council (TÜBİTAK) for their financial support during my study.

I would like to thank my parents for bringing me up and trusting in me. I also thank to my lovely elder brother Oğuzhan and his wife Canan for their encouragement throughout my thesis study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AFDX	Avionics Full Duplex switched Ethernet
BAG	Bandwidth Allocation Gap
CAN	Controller Area Network
CTL	Computational Tree Logic
ES	End System
E2E	End to End
FCFS	First Come First Serve
FIFO	First In First Out
GUI	Graphical User Interface
IP	Internet Protocol
MAC	Media Access Control
MIP	Mixed Integer Program
NS2	Network Simulation 2
QNAP2	Queuing Network Analysis Package
RR	Round Robin
SNMP	Simple Network Management Protocol
TA	Timed Automata
UDP	User Datagram Protocol
VL	Virtual Link

CHAPTER 1

INTRODUCTION

With the development of avionics electronic systems, structure of the airborne data bus, its transmission protocol and its performance are all changed and evolved continuously in recent years. Traditional avionics standards such as ARINC429, ARINC629 and 1553B have low transmission speeds. Furthermore, they have high costs since one function - one computer architecture has been enforced in avionic systems. This increases the number of equipment boxes in system. With the increment in the number of avionic functions it is hard to send information between many avionic devices existing on a typical platform. Avionics Full Duplex Switched Ethernet (AFDX) is proposed to get rid of these problems. AFDX has real-time properties; it is low cost and high bandwidth, and it provides extensive system integration capabilities. Moreover, the data bus weight and quantity is reduced.

AFDX is built around traditional Ethernet with provisions for deterministic behavior and has been developed to provide real-time and reliable performance suitable for avionic systems' demands. 10Mbit/sec, 100Mbit/sec and 1Gbit/sec implementations are available. The media where packets are transmitted in can be either copper or fiber.

End systems, switches and links are three type of elements in AFDX. The AFDX network topology is shown in Figure 1.1. An end system (ES) is a device which is responsible for sending and receiving data from the network. Packet handling, traffic shaping, virtual link scheduling, integrity checking and redundancy management are done by ESs. Traffic policing and filtering are performed by a switch. Packets are forwarded towards their destination ESs in switches. A link provides a connection

between ESs and switches. All connections are full duplex and redundancy is achieved by the duplication of connections.

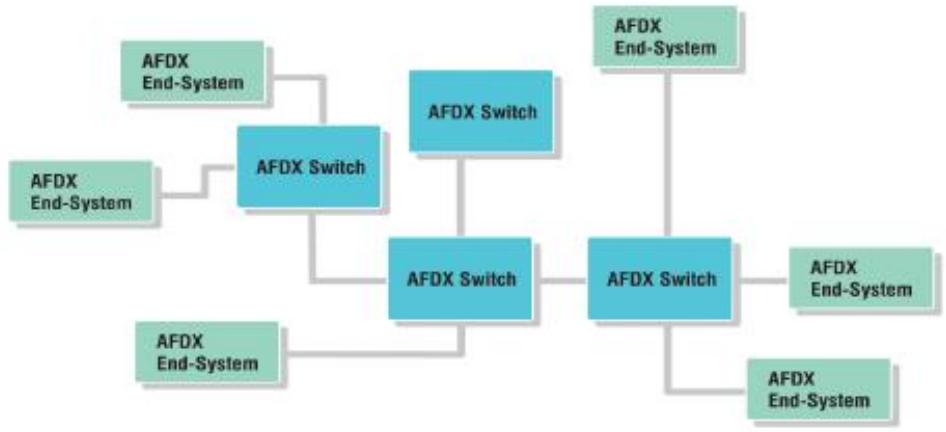

Figure 1.1: AFDX Network Topology [14]

Virtual links (VLs) are utilized for frames exchanged by ESs. A VL is a connection from one source device to one or more destination devices. The connection is unidirectional. Figure 1.2 illustrates how a source ES is connected to one or more destination ESs on an AFDX network. 64K VLs can be defined in AFDX networks.

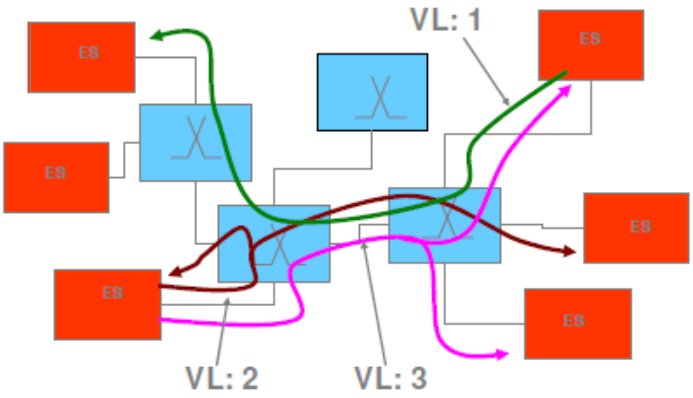

Figure 1.2: Virtual Link Illustration [14]

Following parameters are used for characterization of each VL in an AFDX network. Bandwidth allocation gap (BAG) is utilized for traffic shaping at ESs and

is used for traffic policing at switches. BAG values are specified in milliseconds and can be 2^k where k is an integer in [0, 6] interval. Another parameter, which is very crucial for switches, is the frame size. In order to maintain the forwarding mechanism, a switch should check the maximum frame size. Maximum frame size can be individually configured for each VL and it can be any integer between 64 and 1518 bytes. Frame structure in AFDX network is given in Figure 1.3. The minimum frame size is 84 bytes, which is roughly 6.72 µsec. Transmitting one byte takes 80 nsec in the system. The maximum frame size is 1538 bytes and hence it takes 123.04 µsec within the system [14].

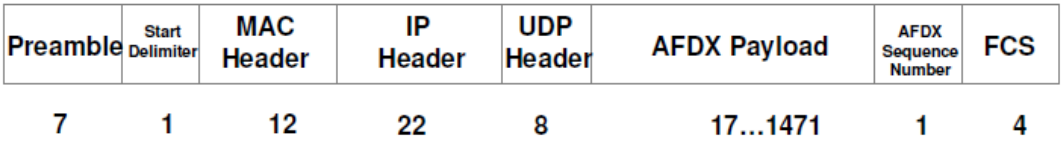

Figure 1.3: Frame Structure [14]

Since AFDX network is upgraded from Ethernet, certain requirements such as reliability, data transmission rate, real-time and low price should all be met. Real-time performance plays a crucial role since an AFDX network is a time-critical network. Therefore, a delay analysis for obtaining the worts case should be examined and an upper bound of delay for each VL should calculated.

1.1 MOTIVATION AND CONTRIBUTION

The worst case E2E delay for VLs can be calculated by using five different approaches. They are network calculus, trajectory approach, mathematical programming approach, network simulation and network modeling [5, 6]. The first three approaches give pessimistic result about the worst case E2E delay due to pessimistic assumptions taken into consideration in these approaches. In the fourth approach, some states may be skipped while calculating the E2E worst case delay so that the obtained result is also not an exact value. For the final approach, timed

automata and UPPAAL software tool are utilized in order to model the system. UPPAAL software tool allows verifying the properties of the network. It also facilitates the usage of different network service disciplines such as FIFO and RR. The motivation behind this thesis work is to construct a tool that generates the state machines of a given AFDX network, which are utilized in UPPAAL software to measure the worst case E2E delay. Another motivation is calculation and comparison of the worst case E2E delay analysis of different approaches mentioned above. Furthermore, usage of different scheduling policies at the output ports of the switches is the final motivation of this thesis.

As for thesis contribution, the developed tool assists in computing the worst case E2E delay for a given AFDX network for its users. With the availability of this tool, users do not spend so much time and effort to find out all scenarios that are candidates for worst case E2E delay. Users can easily obtain state machines that represent ESs and switches in given network topology using this tool. In addition, the comparison for obtained the worst E2E delay using different approaches mentioned above is made to show that network modelling approach gives more accurate result. Moreover, we also applied the RR scheduling algorithm to the output ports of the switches of the given systems in [15] as a second example and compared the use of two service disciplines in terms of worst case delay.

1.2 THESIS ORGANIZATION

The remaining part of the thesis is structured as follows: Chapter 2 gives a brief introduction about the existing methods of computing worst case E2E delay. The advantages and drawbacks of these methods are discussed. Why we chose the modeling approach is argued. Chapter 3 explains the contribution of the present work and introduces the approach used to calculate the worst case delay. Implementation details are given in Chapter 4 and implementation results are provided in Chapter 5. Finally, the thesis is summarized and concluded in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

2.1 METHODS FOR CALCULATION OF WORST CASE END-TO-END DELAY ON AFDX NETWORK

In avionic networks, real-time performance is the most important issue [7]. An upper bound of the delay for transmitting packets in the network is required for certification purposes [5]. Therefore, evaluating whether real-time requirements are met or not plays a crucial role. In other words, the worst case E2E delay on the network should be measured for each flow in the system. Network calculus, trajectory calculation, mathematical programming, simulation and modeling are five major approaches that are used for calculating the worst case E2E delay.

2.1.1 Network Calculus Approach

Network calculus approach determines the upper bound of the delay of a given flow mapped on a given network analytically [6]. To calculate the delay, the concept of arrival curves and service curves should be defined. The number of bits seen on the flow in time interval [0, t] by means of the cumulative function R(t) defines the data flow in this method. Function R is always wide-sense increasing [1]. Wide-sense increasing function α is an arrival curve if and only if for all $s \leq t$:

$$R(t) - R(s) \leq \alpha(t-s) \text{ [1]}$$

Figure 2.1 gives an example arrival curve. The derivative of R(t) is represented as r(t), which is the rate function.

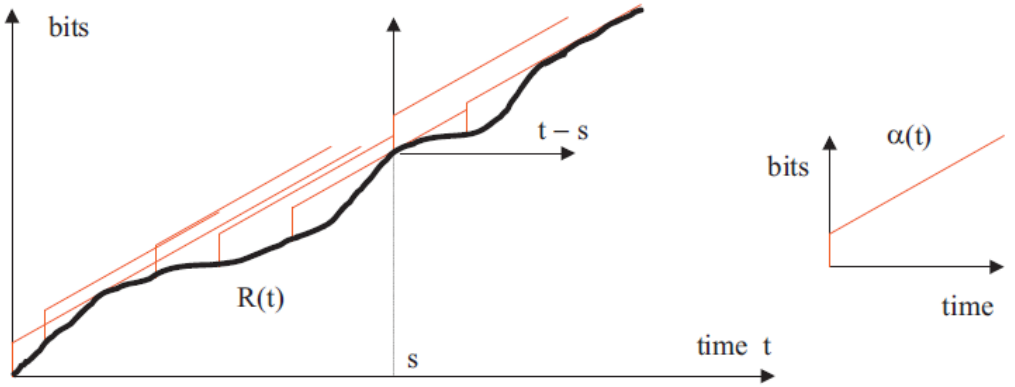

Figure 2.1: Example of an Arrival Curve [1]

To define service curve, a flow through a system S with input and output function R and R^* [1] is considered. The service curve β is wide sense increasing where, $\beta(0) = 0$ and $R^* \geq R \otimes \beta$. Figure 2.2 represents the definition of service curve.

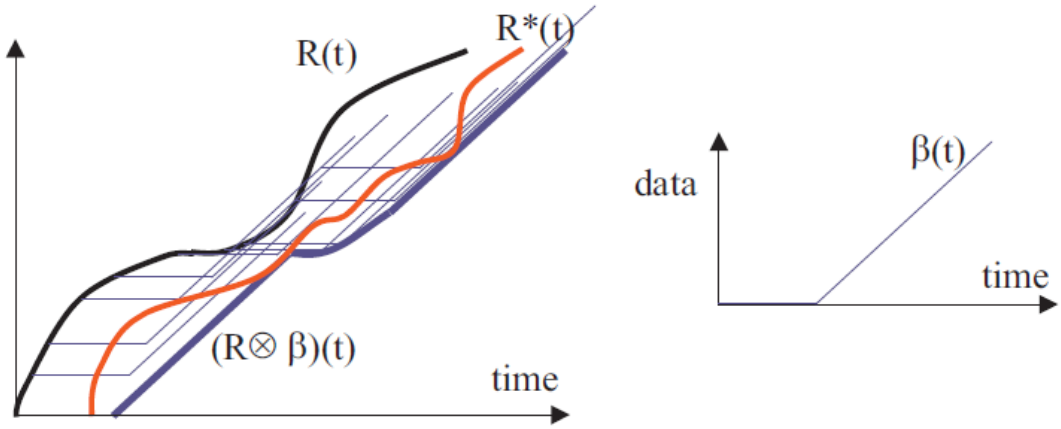

Figure 2.2: Illustration of Service Curve [1]

Two following quantities can be derived by considering input and output functions. The first quantity is backlog, which is the amount of bits that are held in the system. The backlog is the queue length if the system has a single buffer. The backlog at time t is shown as R(t) – R*(t). The second quantity is virtual delay, which the main part of this thesis also deals with. The virtual delay would be counted by a bit that

arrives at a specific time if the other bits are received and serviced before it. This delay value at a specific time is represented as follows:

$$d(t) = \inf \{\tau \geq 0 : R(t) \leq R^*(t+\tau)\}$$

With the definitions of curves that are used as input and output of the system, network calculus gives all bounds for a given systems with service guarantees. The first theorem is Backlog Bound. In this theorem, it is stated that a flow characterized by arrival curve α and passes through a system which offers a service curve β. The backlog for all t satisfies:

$$R(t) - R^*(t) \leq \sup_{s \geq 0} \{\alpha(s) - \beta(s)\}$$

The second theorem Delay Bound assumes a flow again constrained by arrival curve α and passes through a system which offers a service curve β. The virtual delay d(t) for all t satisfies: $d(t) \leq s(\alpha, \beta)$ where $s(\alpha,\beta)$ is the supremum of all values of d(s).

The final theorem Output Flow considers the same assumption with the two theorems given above. In this theorem min-plus deconvolution is utilized to calculate the output. The output flow is calculated as $\alpha^* = \alpha \oslash \beta$.

To sum up, service curve for each flow and departure curve for each output port are considered to evaluate delay analysis of the system. This analysis is done for each flow of each links and results are served for avionics network certification purposes.

2.1.2 Trajectory Approach

In this approach, a frame that traverses each processing node in the system is considered. Unlike previous approach, trajectory approach analyzes the worst case scenario of a packet by considering its trajectory. Then, only possible scenarios are taken into consideration [17].

A distributed system is examined in this approach and the general architecture of the network topology is depicted in Figure 2.3. In the figure, the system is composed of several processing nodes with some links between them. Trajectories are defined by a symbol τ and as an example of a trajectory, τ_1 follows the path $P_1 = \{1, 5, 6, 10\}$. The processing node is never visited again by a packet while transmission in this approach.

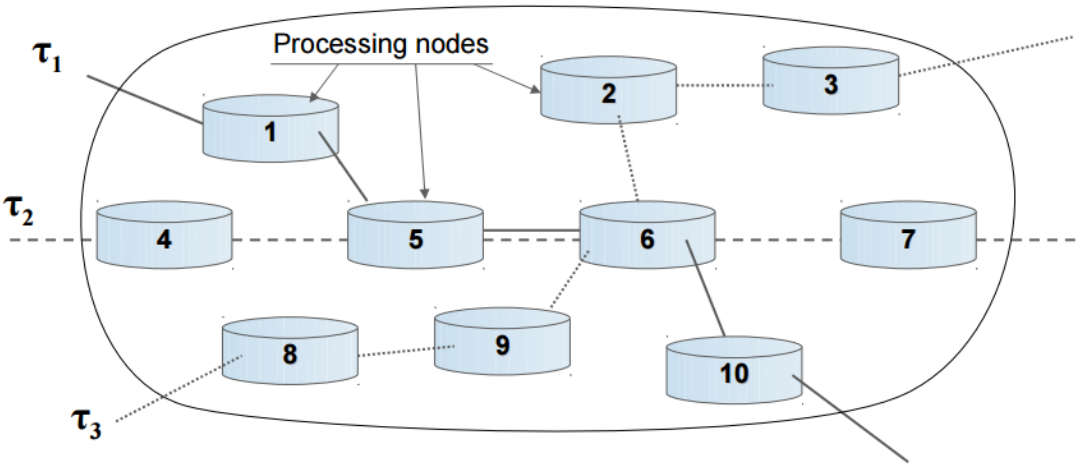

Figure 2.3: Architecture of System in Trajectory Approach [17]

FIFO scheduling algorithm is applied to each flow at processing nodes. At the ingress node there is an inter arrival time between two consecutive packets for each flow. Furthermore, they have an E2E deadline and maximum release jitter. This deadline value is the maximum acceptable response time. Between nodes the transmission time of any packet is limited by L_{min} and L_{max} values. The time spent for transmission delays on links and time spent at each crossed nodes are added to calculate the E2E delay of a packet.

The busy period is main concept in the trajectory approach. A busy period level L is a time interval [t, t') such that start and end time are both idle. All the packets generated before an idle time t are processed at this idle time. To measure the E2E delay using trajectory approach, a packet is considered with its flow and transmission time. Then, trajectory approach identifies the busy period and each delay on the nodes crossed by the packet. This decomposition enables the calculation of the latest starting time of the packet on its last node. The worst case E2E response time of the flow can be computed recursively at each processing node.

2.1.3 Mathematical Programming Approach

To calculate the worst case E2E delay of AFDX network, mathematical programming approach can be utilized. In this approach, given an AFDX network is converted to a mathematical model [29]. In other words, certain mathematical constraints are defined to calculate the worst case E2E delay.

This approach is actually a maximization problem. It maximizes the subtraction operation. The arrival time of a frame at the source node is subtracted from the arrival time of that frame at the destination node. However, certain constraints should be defined in this maximization problem. These constraints are related to frame transmission in the system.

For the first rule, if a frame is transmitted from node *s* to node *d*, then the arrival time of a frame at node *d* is greater than the arrival time of a frame at node *s*.

$A_d \geq A_s + e_{s,d}$, where $e_{s,d}$ is the transmission time between nodes *s* and *d* [29].

If two frames are transmitted over the same link, then the relation between the arrival times of these two frames at nodes constructs the second rule, which is:

$$A_s^i \leq A_s^j \rightarrow A_d^i \leq A_d^j, \text{ where i and j are frames [29].}$$

With the second rule, third rule can be defined clearly.

$$A_s^i \leq A_s^j \rightarrow A_d^i + e^j_{s,d} \leq A_d^j \text{ [29]}$$

To use MIP, a binary variable X is added to third rule and this binary variable is used for which frame comes first at a specific node. X is defined as following:

$$X^{i,j} = \begin{cases} 0 & \text{if } i \text{ arrives before } j \\ 1 & \text{if } i \text{ arrives after } j \end{cases}$$ [29]

Another binary variable F is used to represent whether a packet arrive at a same node immediately after arrival of another packet. F is defined as following:

$$F_d^{i,j} = \begin{cases} 1 & \text{if } i \text{ arrives at } d \text{ immediately after the arrival of } j \\ 0 & \text{otherwise} \end{cases}$$ [29]

With the binary variables and rules mentioned above, the maximization problem can be derived. The maximization problem with its constraints is given below:

$$maximize : A_{d_o}^i - A_{s_o}^i$$

subject to:

$$\begin{aligned}
&A_d^i \geq A_s^i + e_{(s,d)}^i && \forall (s,d) \in E, \forall T_i \in C_{(s,d)} \\
&X_s^{i,j} + X_s^{j,i} = 1 && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)} \\
&A_s^i - A_s^j \leq M(X_s^{i,j}) && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)} \\
&A_d^i - A_d^j + e_{(s,d)}^j \leq M(X_s^{i,j}) && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)} \\
&F_d^{i,j} \leq X_s^{i,j} && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)} \\
&A_d^i - A_d^j \leq M(1 - F_d^{i,j}) + e_{(s,d)}^i && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)} \\
&A_d^i \leq A_s^i + e_{(s,d)}^i + M \sum_{j \in C_{(s,d)}} F_d^{i,j} && \forall (s,d) \in E, \forall T_i, \in C_{(s,d)} \\
&A_s^i \geq 0 && \forall (s,d) \in E, \forall T_i, \in C_{(s,d)} \\
&X_s^{i,j}, F_d^{i,j} \in \{0,1\} && \forall (s,d) \in E, \forall T_i, T_j \in C_{(s,d)}
\end{aligned}$$ [29]

The problem can be solved by utilizing tools such as IBM CPLEX solver.

2.1.4 Simulation Approach

Network simulation is another approach for performance research of AFDX network. Unlike two approaches given above, the network characteristics can be clearly followed by simulation approach. This approach can also identify the connetion between various modules in a complex model. Moreover, other performance parameters, comparison and the analysis of the system, are provided [8]. To perform simulations, several platforms such NS2 (Network Simulation 2) and QNAP2 (Queuing Network Analysis Package) can be used.

In order to show how these platforms operate NS2 is analyzed in detail in [8]. NS2 is a tool which is mainly used for the simulation of network architectures. AFDX simulation model is illustrated in Figure 2.4. As was mentioned earlier and also seen in figure, a switch is responsible for filtering and policing operations. End systems provide system configuration, packet handling and VL scheduling. In Figure 2.5 the data creation, store-and-forward switch transmission, end system transmission, the direction of flow and end system reception blocks are illsustrated. The flow chart of sending data from an ES and transmitting data from switch algorithms are presented in Figures 2.6 and 2.7. By providing the AFDX network topology with its ESs and switches, the NS2 platform then calculates E2E delay for each VL defined in the system.

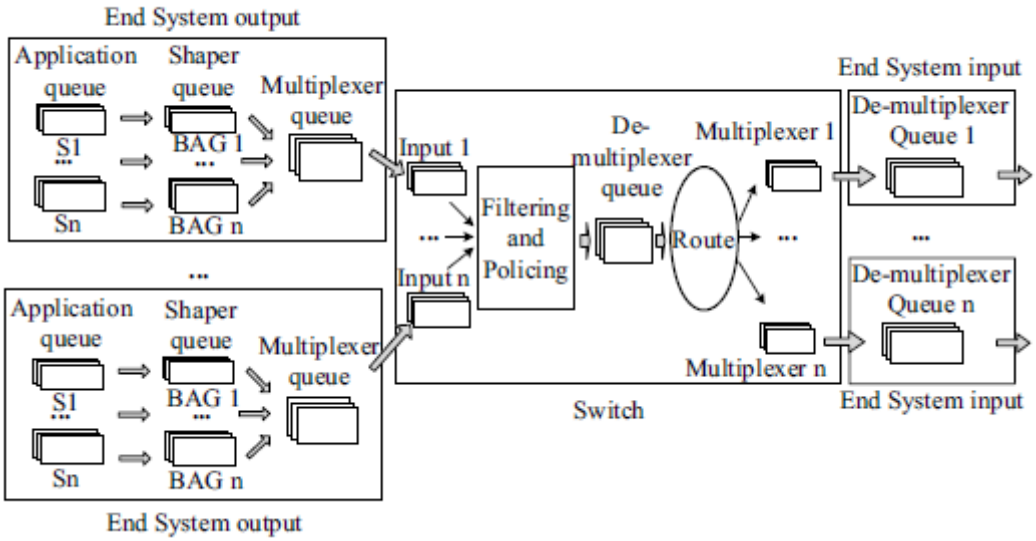

Figure 2.4: AFDX System in Simulation Model [8]

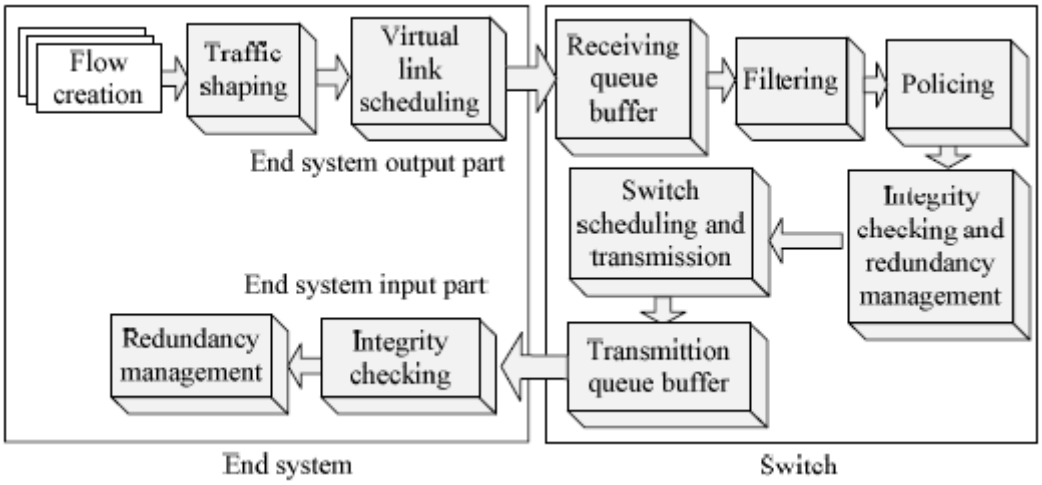

Figure 2.5: AFDX Network Simulation System [8]

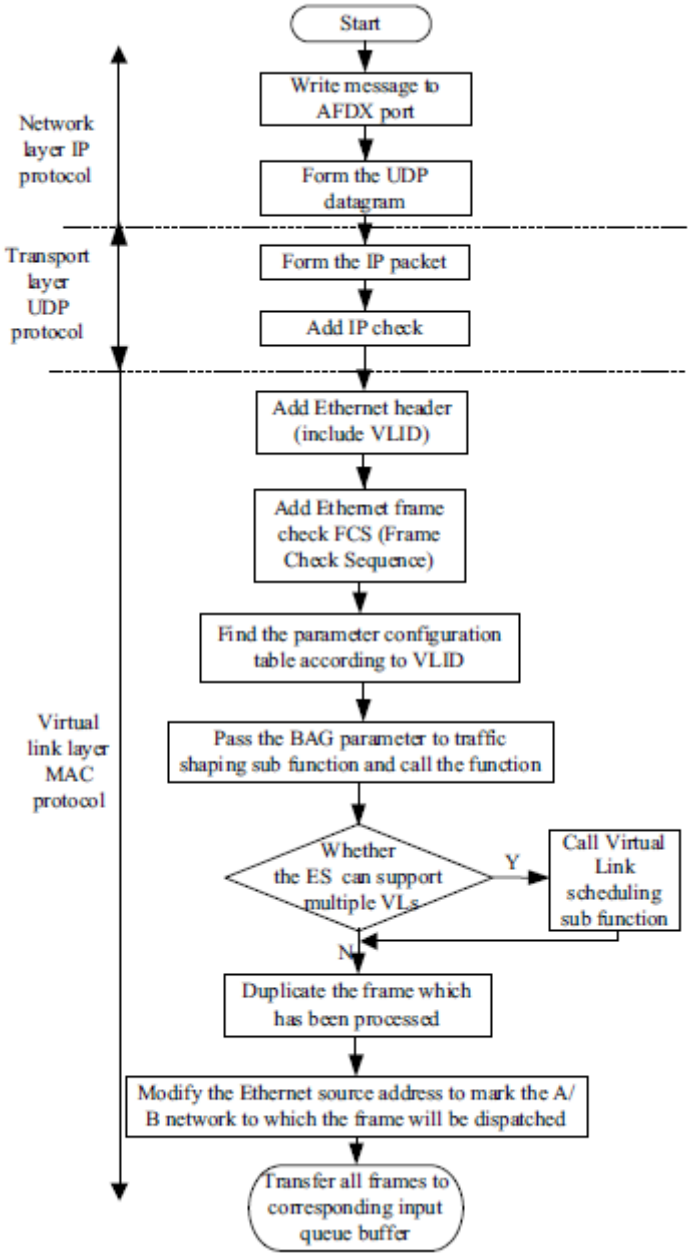

Figure 2.6: Sending Data Process in End System [8]

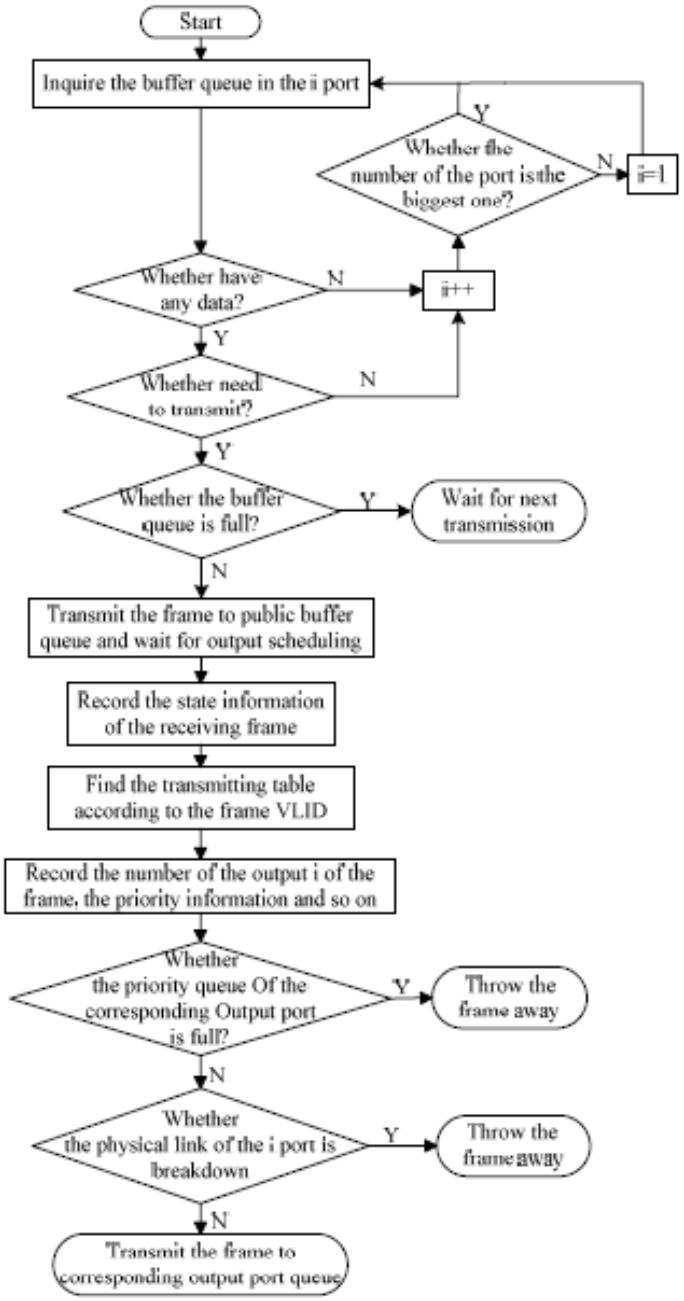

Figure 2.7: Dispatching Data Process in Switch [8]

2.1.5 Network Modeling Approach

Worst case E2E delay values obtained by network calculus, trajectory approaches and mathematical programming approach are observed to be not reached in real systems [2] [4] [5] [6]. The main reason for this is making pessimistic assumptions on the arriving flows. In simulation approach, most possible scenarios are considered and then the worst case E2E delay is computed. However, it can still not be taken as the best approach since it is possible to miss rare events in the system.

In [2], network calculus and simulation approaches are compared using a given AFDX network topology. The network is depicted in Figure 2.8. This network architecture is composed of 8 switches and numerous VLs and the test configuration is provided by Airbus. At the output port of switches, FIFO buffers are used. As was mentioned earlier, the system should have one source ES and one or more destination ESs. In [2], the input port of switch S1 is taken as the source and the outputs of switches S3, S4, S7 and S8 are the destination ports. The delay calculations are done according to this structure.

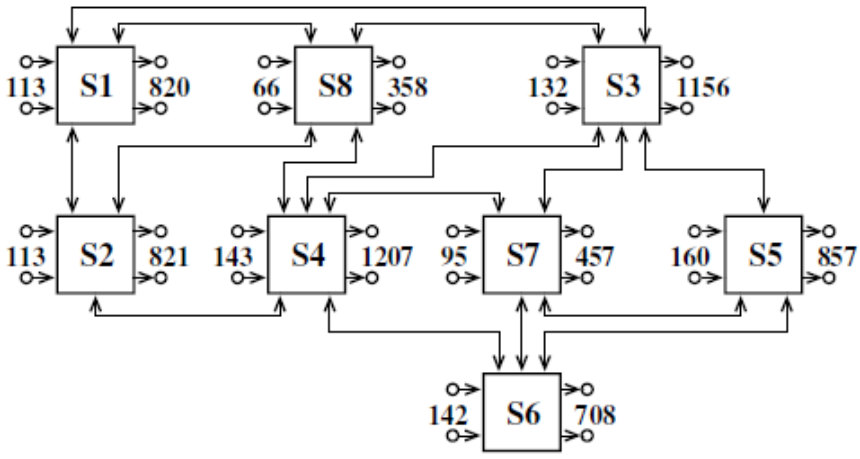

Figure 2.8: AFDX Network Test Topology [2]

The configuration provided by Airbus includes 964 VL and 6384 paths. The number of VL between switches S and D is given in Table 2.1. Following table represents the number of VL dispatching among BAG values and frame lengths. The load of

the system is shown in Figure 2.9. For each possible load the number of physical link is given in the graph.

Table 2-1: Number of VL between switches S and D [2]

S \ D	1	2	3	4	5	6	7	8
1		71	78					34
2	72			77				34
3	90			212	35		42	52
4		97	134			37	35	48
5			80			72	64	
6				82	61		52	
7			52	47	59	67		
8	51	45	43	52				

Table 2-2: BAG Values and Frame Lengths [2]

Bag (ms)	Number of VL	Frame length (bytes)	Number of VL
2	20	0-150	561
4	40	151-300	202
8	78	301-600	114
16	142	601-900	57
32	229	901-1200	12
64	220	1201-1500	35
128	255	> 1500	3

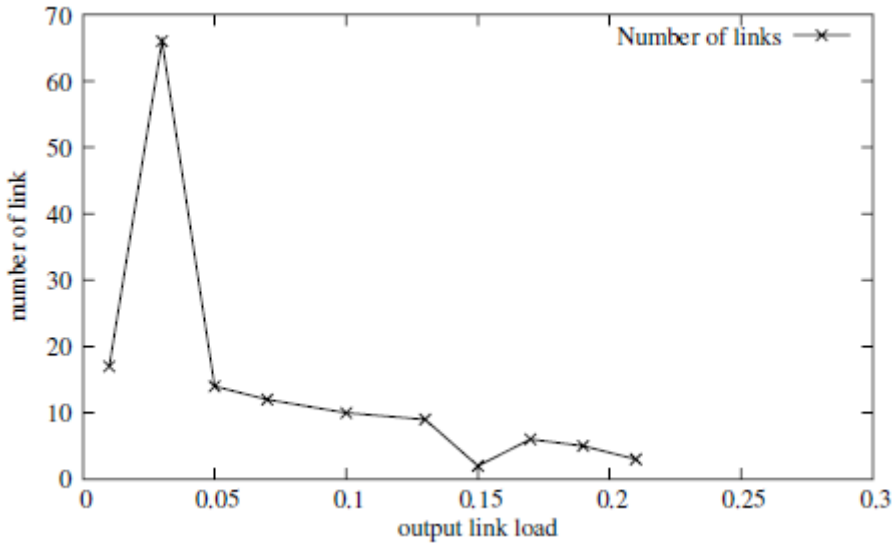

Figure 2.9: Load of the System [2]

The following figure compares desired delay value obtained by two approaches network calculus and simulation. The delay value is obtained under certain circumstances. The generation of frames is periodic and the specified BAG values are used for periocity. In Figure 2.10, E2E delay ratio of two approaches is computed for each VL path. Null phasing and random phasing are related to synchronization of frames. In null phasing, all VLs send their first frame at the same time. For the other, first frames are generated randomly. Using the figure, it can be stated that the ratio of simulation delay to network calculus delay ranges between 5% and 40% for most of VL paths. When VL paths cross a single switch then the ratio is at least 70% [2]. Furthermore, as the number of VL path increases, network calculus approach provides a more pessimistic result [2].

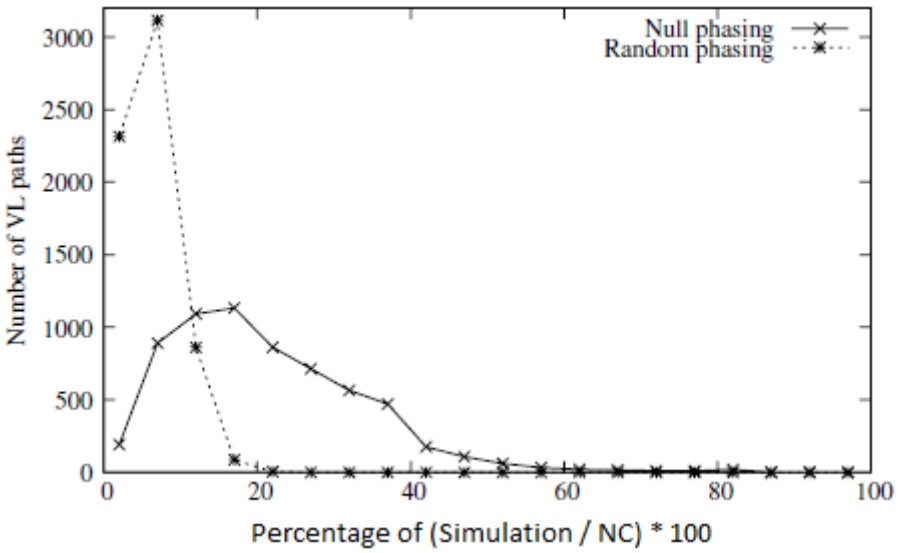

Figure 2.10: Comparison of Delay Bounds for Simulation and Network Calculus Approaches [2]

Model checking on the other hand provides more realistic results for AFDX network delay analysis. It stands on timed automata. All possible states are explored in the system by this method and then the exact upper bound of delay is determined. However, model checking approach cannot be utilized for larger AFDX network for example the ones including more than 10 VLs since it implements an exhaustive search of all possible scenarios. In order to increase the size of the configurations, search space should be limited drastically. This limitation is handled in [5]. In [5], it is stated that implementing ESs with their local clock enables us to cope with larger networks. Equivalently, VLs of each ES are scheduled locally.

In the present thesis work, model checking approach, which is based on timed automata and UPPAAL, is utilized to perform worst case E2E delay calculation. The two terms mentioned, namely “timed automata” and “UPPAAL” are briefly introduced in the following sections.

2.2 TIMED AUTOMATA

Regular techniques for model checking do not give an accurate modeling of time and therefore they are not suitable for real-time systems to be analysed [18]. So as to model the behavior of real systems timed automata were introduced. They have first been proposed by Alur and Dill [2], [19]. Unlike automata, timed automata include timing constraints in state transitions.

Systems in timed automata are modeled by state transition graphs where event symbols are used to identify transitions. A transition system is a tuple (Q, Q_0, Σ, δ), where Q is a set of states, Q_0 is a set of initial states, Σ is a set of events and $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions [18]. For instance, a transition (q, α, q') in δ is written as $q \; q'$. The system starts in an initial state and if a transition $(q \; q')$ occurs then the system changes its state from q to q' on event α.

Finite graphs in automata are improved with clock constraints to express the behavior of the system with timing constraints. The clocks in systems are real-valued. The following figure illustrates an example of a timed automaton. In this figure, it is seen that state *s* has no invariant constraint. Having no invariant constraint means that the system can stay in the corresponding state *s* for an arbitrary amount of time and until event *a* occurs. Furthermore, there is a single clock *x* in the system and it gets reset to 0 when the transition from *s* to *s´* occurs. The invariant in location determines how long a system stays in that location and the system must leave that location before the invariant is violated. However, in order to execute action *b* there exists another timing constraint on transition from *s´* to *s*. Thus, action *b* can occur if the time value is in between 1 and 2.

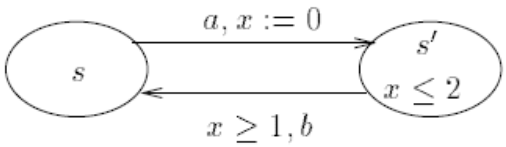

Figure 2.11: Timed Automaton Example [18]

Timed automata can have more than one clock. This allows modelling multiple concurrent delays in the system. These clocks can be utilized in both states and edges of the system.

Semantics and syntax of a timed automaton is given below: A timed automaton A is a 6-tuple $(L, L^0, \sum, X, I, E)$ where in an order;

- L is a finite set of states,
- $L^0 \subseteq L$ is a set of initial locations,
- $\sum$ is a finite set of events,
- X is a finite set of clocks,
- I is mapping that labels each location s in L with some clock constraint in $\phi(X)$,
- $E \subseteq L \times \sum \times 2^x \times \phi(X) \times L$ is the set of transitions,
- $\phi(X)$ is clock contraints [18].

In this thesis work, timed automata are used to model a given AFDX network within the UPPAAL framework. The ESs and switches are constructed as automata in the developed tool. With the synchronization property of UPPAAL, following the construction of the whole system, worst case E2E system delay is calculated easily.

2.3 UPPAAL TOOL

UPPAAL developed by Uppsala University and Aalborg University is a tool for authentication of real-time systems [20] [25]. The first version was released in 1995 [20]. This tool enables users to investigate the behavior of their systems in terms of state transitions and to simulate and analyze the constructed systems. In other words, it is used for modeling, simulating and verifying a given system. The architecture of the tool is given in Figure 2.12. States and edges of timed automata are established by a graphical user interface (GUI). The properties or functions that belong to timed

automata are coded in C++ [22] [23]. The constructed system can be saved in *xml*, *xta* or *ta* formats.

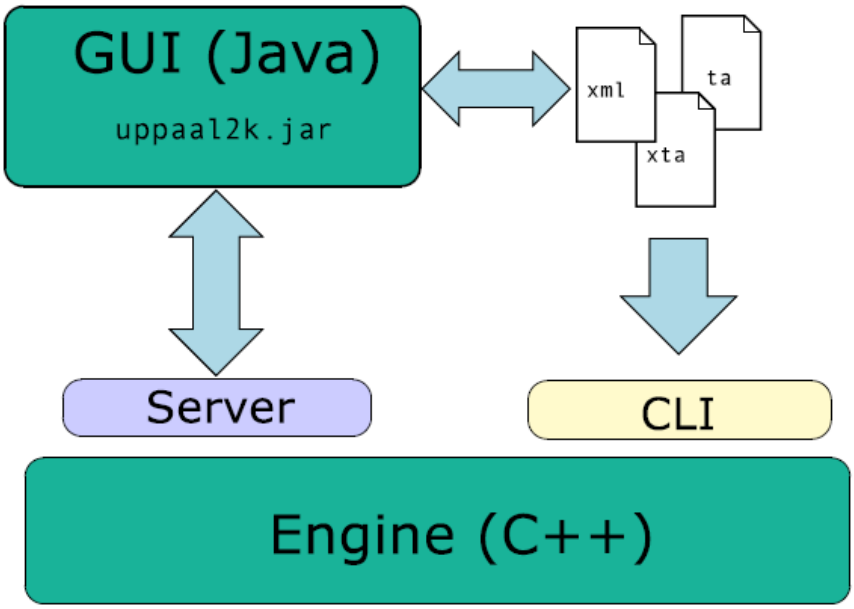

Figure 2.12: Architecture of UPPAAL Tool [22]

In UPPAAL, a network of several timed automata is modeled in parallel. Bounded discrete variables that belong to states extend the model and these variables are utilized as in a programming language. Figure 2.13 shows timed automata modeling in UPPAAL.The figure presents two automata related to each other. In this model, there is a user who is responsible for pressing a button and turning a lamp on or off. According to how fast the user presses the button, brightness of the lamp changes. Two fast pressings make the lamp brighter. This is achived by the help of a clock variable *y* and a channel called *press.* The meaning of ! and ? are explained in the following paragraphs. Another fact that can be noticed from this figure is that the two automata are processed in parallel and this is done by synchronization signal defined on an edge.

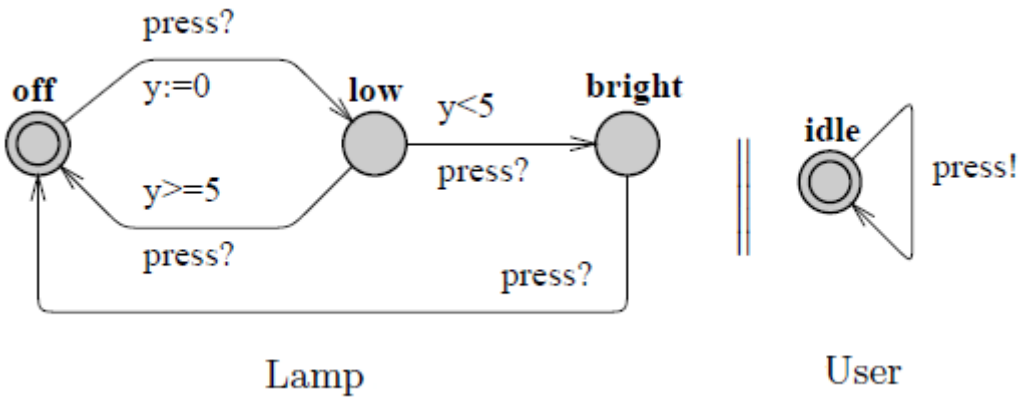

Figure 2.13: Lamp Example of UPPAAL Tool [20]

Timed automata are extended in UPPAAL with some additional features some of which are depicted in Figure 2.14. A short summary of these properties are given below:

Templates are used as timed automata with a specified set of parameters that can be of any type. There can be more than one templates working synchronously in the project.

Constants are integer values used in systems that cannot be modified.

Channels are defined on edges and are decleared as *chan c*. Characters ! and ? are added at the end of these channels for synchronization of templates. ! means output and ? means input. Adding these characters change the syntax of the channel in the tool as *broadcast chan*. Another channel that prevents delay in the transition action is declared as urgent broadcast chan.

Locations such as circles in Figure 2.13 represent the states of a timed automaton. In UPPAAL there are three types of locations. First one is the *initial* state that the system starts from. *Urgent* location is the second one where time is not allowed to pass in. *Committed* location is the last one that is more restrictive than the urgent location. It cannot delay the system and next transition must involve an outgoing edge.

Arrays can be used for constants, integers, channels and clocks.

Expressions in UPPAAL are defined in both locations and edges. *Select*, *guard*, *synchronization*, *update* and *invariant* are possible labels of expressions in this tool.

Select label contains a comma seperated list of *name : type* expressions where *name* is a variable name and *type* is a defined type [20].

Guard is an expression that sets a condition on transition between states. It is side-effect free and can evaluate only clocks and integer variables.

Synchronization expressions provide communication between automata in the system.

Update expresssion is a comma seperated list that must only refer to clocks, integer variables and constants. It may also call functions.

Invariant expression can aonly be seen in locations of the automata. It actually allows how long a system stays in a state where invariant is defined.

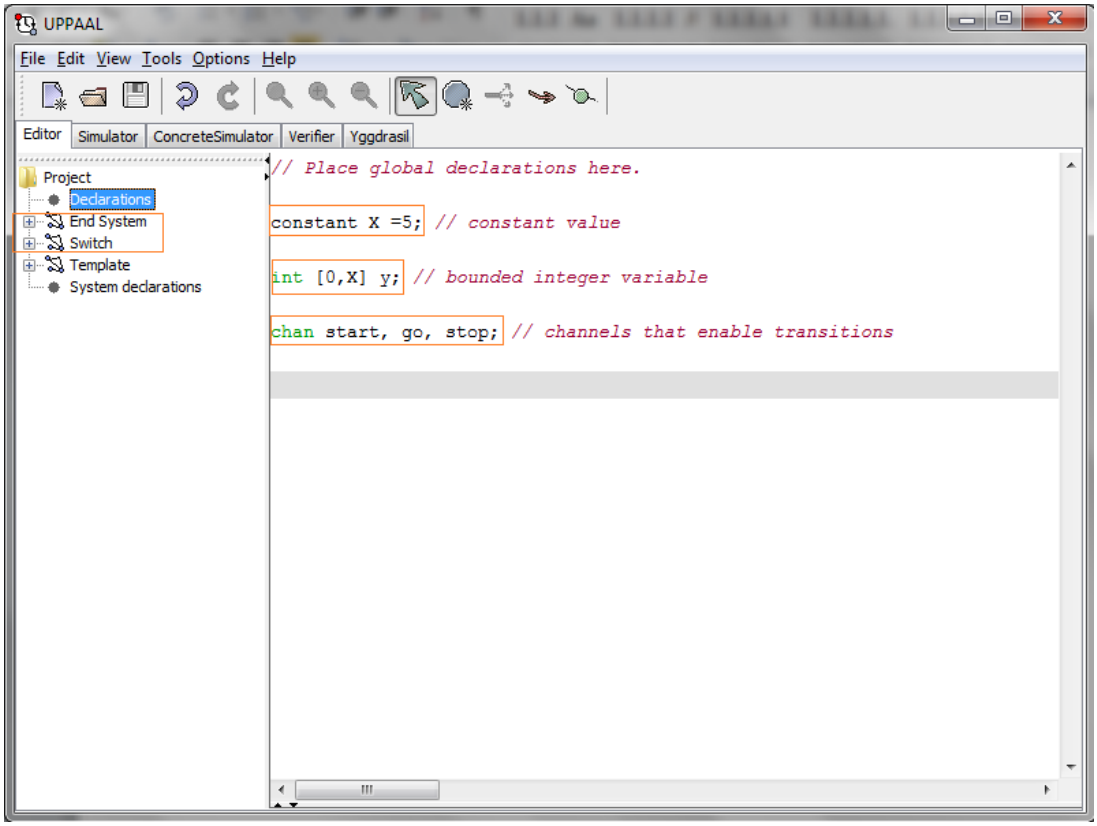

Figure 2.14: Features of UPPAAL Tool

There are four drawing tools *Select*, *Location*, *Edge* and *Nail*. To construct the system these tools are utilized. *Select* tool is used to select, move, modify and delete elements. *Location* tool adds new states to the system. *Edge* tool simply connects two states. *Nail* adds new nails to an edge.

Figure 2.15: Drawing Tools in UPPAAL

The most crucial issue that should be considered carefully while using the tool is how clock behaves in the system. In order to figure out the behavior of the clock better, a simple example is taken from [20]. There is a simple reset system depicted in Figures 2.16 and 2.17. In the system, the clock is counted and after some amount of time the clock gets reset. In Figure 2.16(a) transition occurs if the clock value is greater than 2. It is satisfied with the guard expression $x \leq 2$. However, the exact time of the transition is not known. This can be seen from Figure 16(c). Nevertheless, adding invariant expression $x \leq 3$ to the *loop* state sets an upper bound for this transition time. Figure 2.17(b) shows that the clock value is real-valued and the transition can be shaped with the expressions of *guard* and *invariant.* In order to change states whenever the value of the clock is an integer, the *guard* and *invariant* expressions should take the same value.

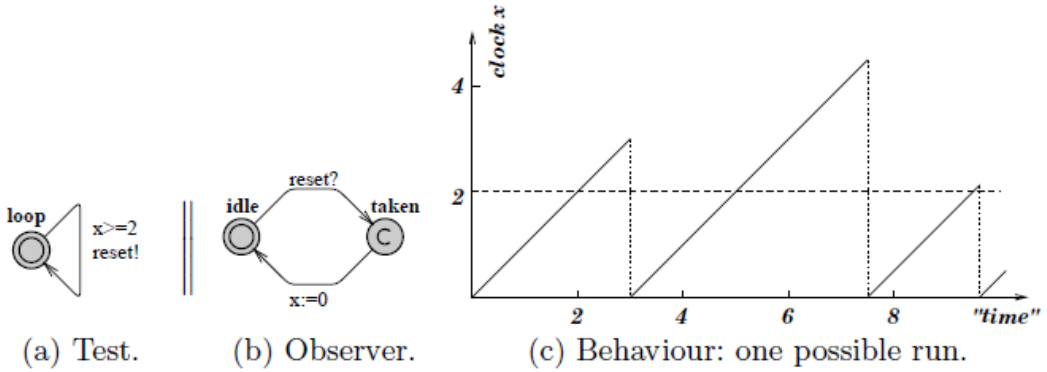

Figure 2.16: Reset System Example without an Invariant [20]

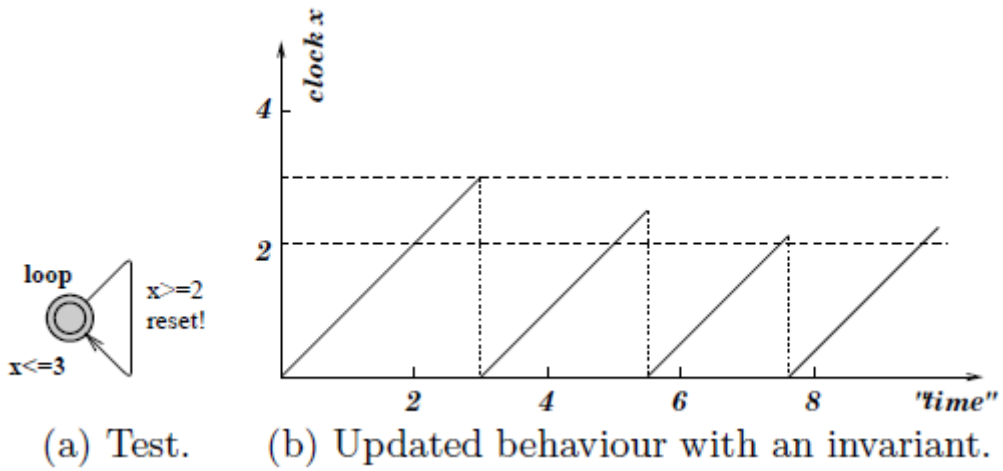

Figure 2.17: Reset System Example with an Invariant [20]

All of the properties, especially the one related to clocks, of UPAAL tool are used carefully to construct the AFDX network model studied in this thesis.

CHAPTER 3

RELATED WORK

3.1 CALCULATION OF WORST CASE END-TO-END DELAY IN AFDX NETWORKS USING MODEL CHECKING

Utilizing model checking approach to calculate the worst case E2E delay is not a new concept. However, it is lightly different from CAN/Switched Ethernet structure. It searches all possible states of the given system and then determines a definite worst case E2E delay.

A network can either be constructed artificially or be a part of an industrial configuration in order to calculate an exact worst case E2E delay. A typical industrial configuration consists of more than 1000 VLs and 130 ESs, which cannot be handled by software tools [15]. After fixing the network topology, the system is modelled. In other words, timed automata are constructed separately for each ES and switch that exists in the system. While doing this, all parameters of VLs should be well defined. The periodicity of frames generated from each ES must be determined. In other words, packet generating functions should be modelled. For each switch, scheduling of their output ports must be arranged. Generally, FIFO scheduling algorithm is utilized but in this thesis RR scheduling algorithm is also considered.

Synchronization between each automaton should be considered which we believe is the most critical part in the model for worst case E2E delay calculation. The time to send packets from each ES and switch affects worst case E2E delay directly. The network should be loaded as mush as possible so that the worst case E2E delay is correct and precise.

3.2 TOOL SUPPORT FOR CALCULATION OF WORST CASE END-TO-END DELAY

A tool is offered for the calculation of worst case E2E delay in this thesis work. The purpose of this tool is to provide state machines of each ES and switch to UPPAAL software so that the desired delay value is measured easily. In this tool, the user is confronted with several forms. In these forms, configuration data of the given network is input from the user. Configuration data includes number of VLs of each ES, BAG values of each VL, packet size and offset values between VLs. According to this user data, the tool generates templates with the necessary features. These templates can be imported to UPPAAL software afterwards. As the final step synchronization signals to state machines should be modified by the user.

To compute the worst case E2E delay, user should then check the query supplied in UPPAAL. When the check box is clicked in the *Verifier* section in UPPAAL, UPPAAL provides the result whether the query is verified or not after some time.

3.3 SCHEDULING ALGORITHMS USED AT OUTPUT PORT OF SWITCHES

Two algorithms, namely FIFO and Round-Robin are used in calculating the worst case E2E delay in the AFDX networks in this thesis.

3.3.1 First In First Out

FIFO is a scheduling algorithm that is used for managing a data buffer, in which the oldest entry to the system is served first. This is a queuing system tha is similar to first-come first-served (FCFS) behavior. The following figure illustrates FIFO process. In AFDX network, FIFO algorithm is utilized at the output ports of switches. Packets arrive from each ES to become a part of output queues in switches and then they are served according to their entry order.

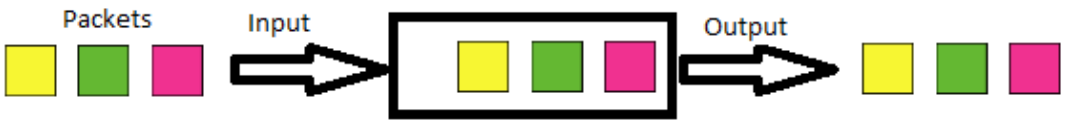

Figure 3.1: Sample of FIFO Algorithm

3.3.2 Round-Robin Scheduling Algorithm

RR is another scheduling algorithm used in computer networks. In this algorithm time is divided into equal portions and in each portion of time different processes are handled in circular order. The algorithm is simple, easy to implement and starvation-free. In AFDX networks, if RR scheduling is used, each ES has its own queue on each port because switches serve one packet from each queue of ESs within an assigned time slice. There is no need to have separated a queue of different ESs in the FIFO case. Figure 3.2 illustrates RR algorithm. In thesis work, RR algorithm is also taken into account to see its effect on worst-case delay.

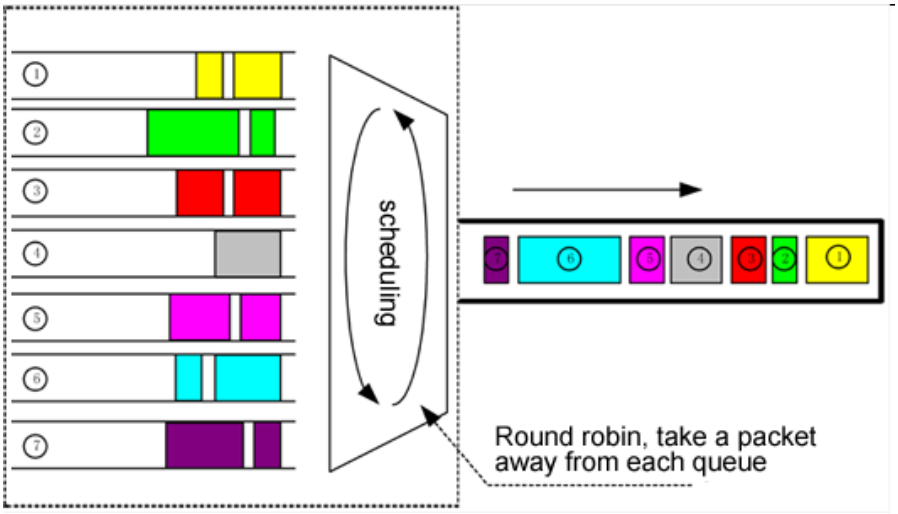

Figure 3.2: Sample of RR Algorithm

CHAPTER 4

IMPLEMENTATION

This chapter includes the implementation details of the tool we developed for the calculation of exact worst case E2E delay utilizing the network topology given in [15] as a sample input network for the tool. This delay value is calculated accordingly.

4.1 WINDOWS FORM APPLICATION TOOL

The aim of this tool is to produce different templates in *xml* format that are going to be imported into UPPAAL software. These files keep information about each ES and switch in the given network, more precisely, data such as number of VLs, offset value of VLs, etc. Furthermore, the synchronization issue which is focused in the next session is considered.

When the application runs, user is asked to input information about the ES under study. The information to be provided includes the name of the ES, the number of VLs that ES has, the offset values of each VL and the name of the VL that is going to be used for the calculation of the worst case E2E delay in the system. As can be seen from Figure 4.1 *Write Template* button creates a template for the ES under study. When this button is clicked, a new window is opened and the file is saved as shown in Figure4.2. *Next* button allows user to use the next form.

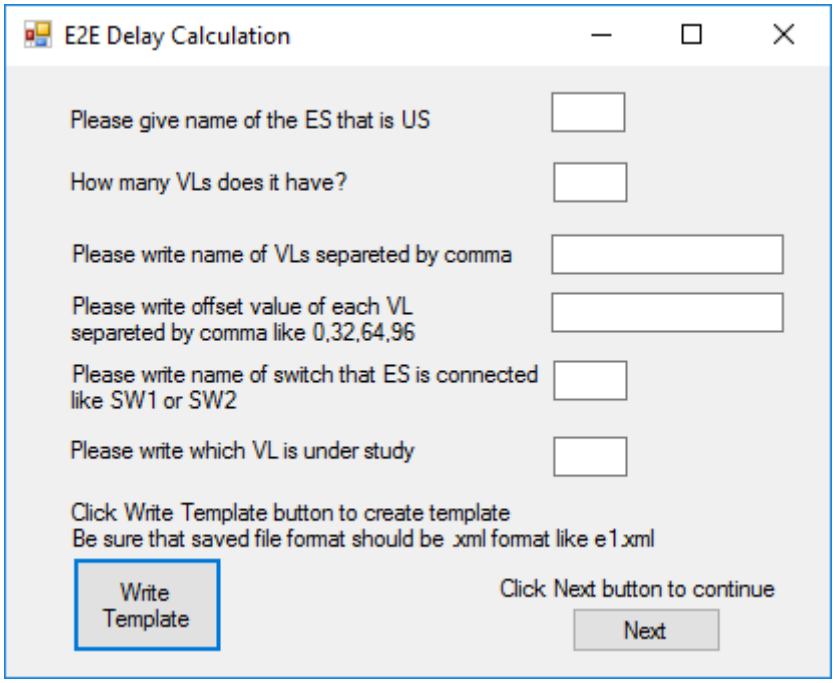

Figure 4.1: Input for ES under Study in Windows Form Application

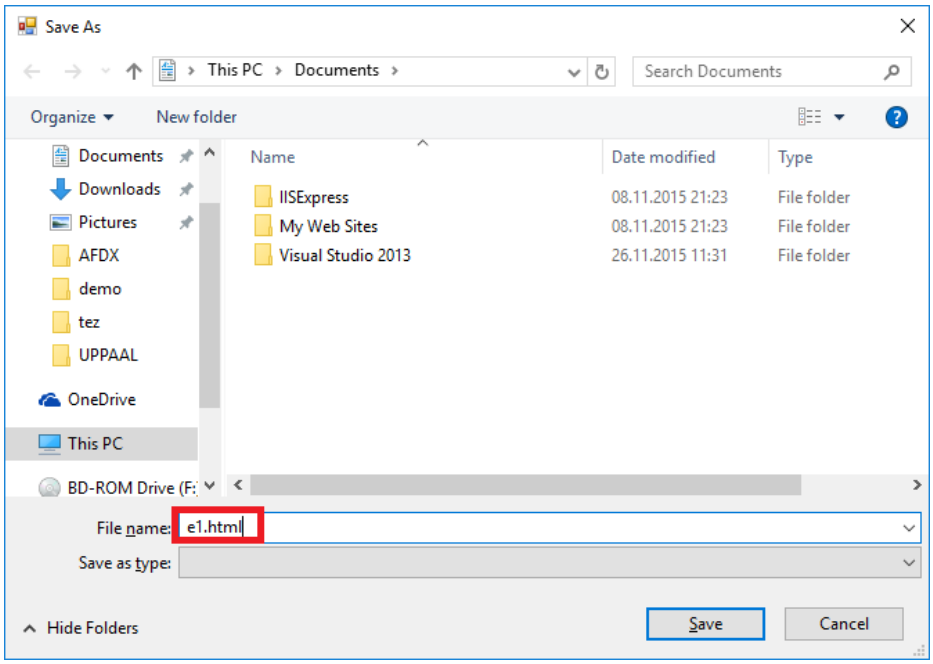

Figure 4.2: Save Template File for Network Components

When the next button is clicked, input data that belongs to ESs, which areconnected to same switch as ES under study is extracted from the user. The same procedure is applied, except that if ES under study has more than one neighbor, the user should click *Write Template* button to create another *xml* file for other ESs. The associated window is given below.

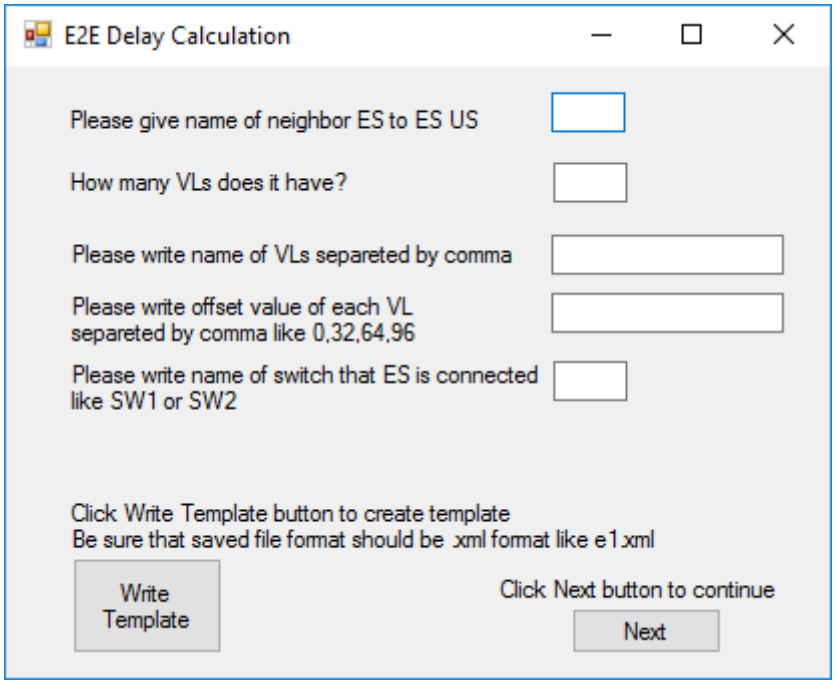

Figure 4.3: Data Entry for ESs Neighboring to ES under Study

The next form is used for gathering information about other ESs in the system. Similar to the previous one *Next ES* button is used if more than one ES exists in the system. This window is shown in Figure 4.4.

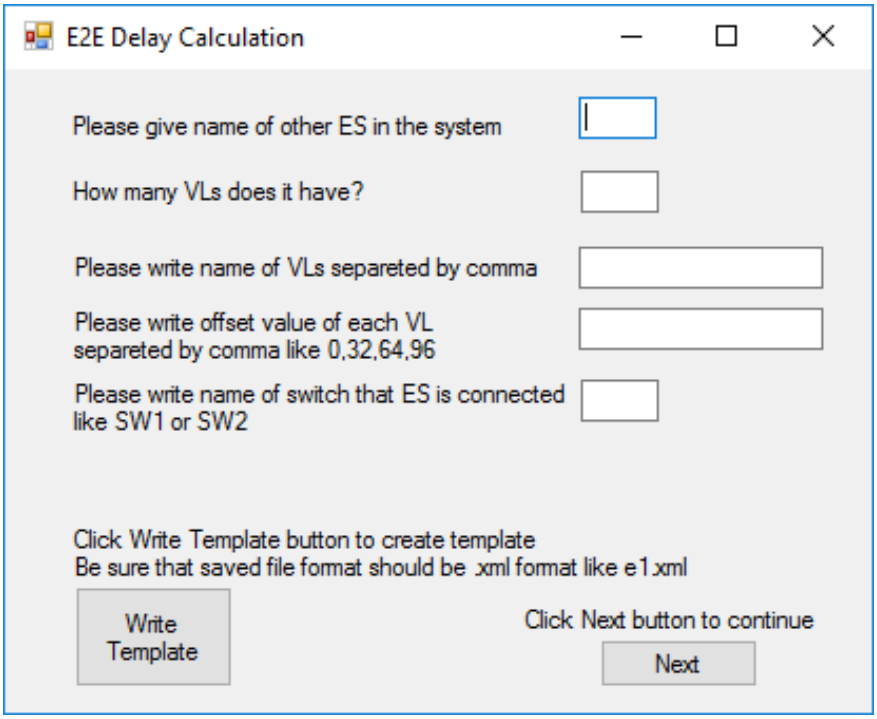

Figure 4.4: Other ESs in the AFDX Network

If the system has sporadic VLs, user is required to provide information about them also as at the next step. For sporadic VLs only BAG value is needed to create a template. Figure 4.5 presents the associated input window for this step.

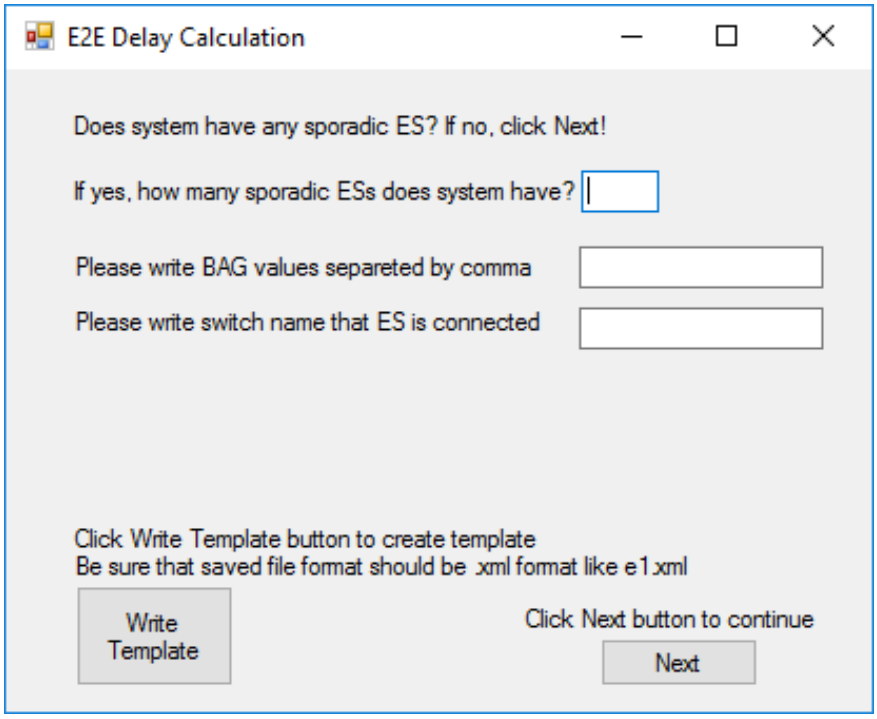

Figure 4.5: Sporadic VLs in AFDX Network

When the templates of ESs within the system are created, the data related to switches are input from the user. At this stage the name of the parent switch of the topology is read from the user. This follows inputing the name of parent switches of the switches specified in previous windows. Name of switches remained in the system are provided at the bottom of this window. Figure 4.6 shows this input process.

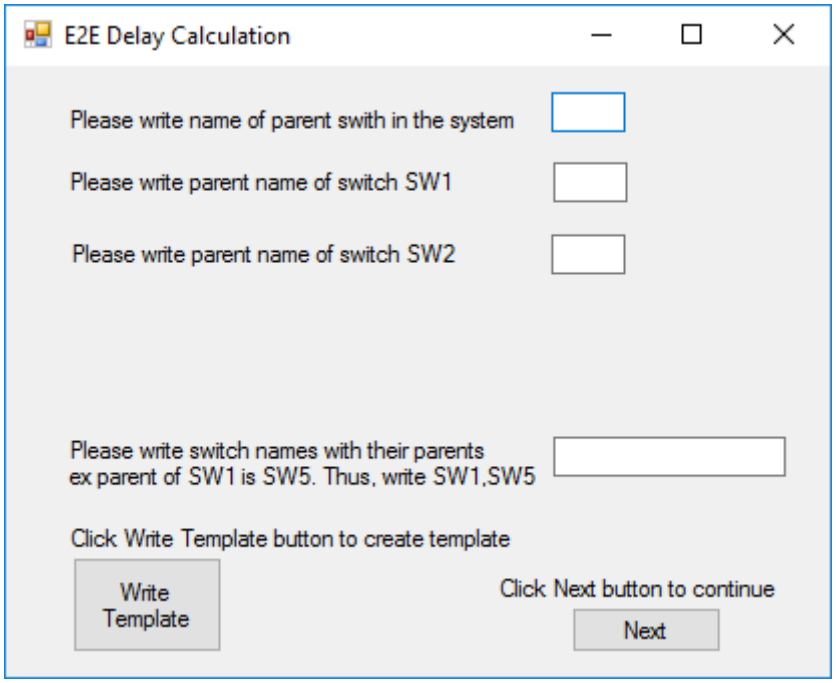

Figure 4.6: Switches in AFDX Network

At the final stage the tool asks user to enter the packet size and click a button. The packet size is taken as a constant but in UPPAAL it can be changed. When the user clicks the final "write system template" button, the tool generates two more templates, which are used for integer clocks in measuring delay value.

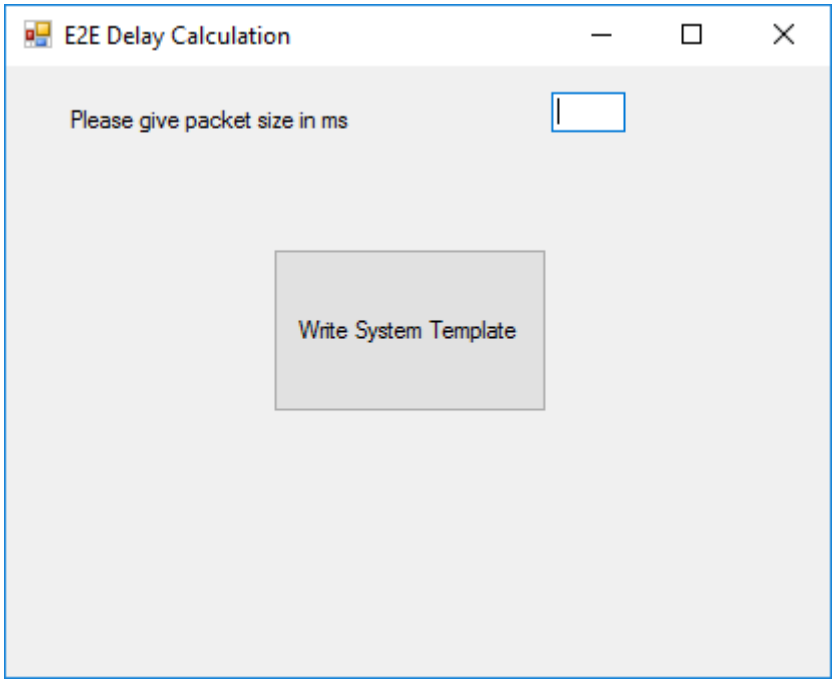

Figure 4.7: Create System Template

After creating these templates, they are imported into UPPAAL from where they are stored. To achieve this, first, the templates created in the final input stage should be added as a system in UPPAAL. Then, the other templates should be added one by one by using *Import Template* choice in *File* menu of UPPAAL. When all *xml* files are imported, synchronization signals should be modified as described in the next section. Following this modification, user selects the query in the *Verifier* section and clicks the *Check* button to calculate the worst case E2E delay.

4.2 AFDX NETWORK ARCHITECTURE

In AFDX ESs are connected with physical links. In this network, there is no global clock, meaning that the system works asynchronously. Transmitting and receiving processes are executed throughout VLs, which are unidirectional connections from one source to one or more destination points in the system. The routing of the paths is known and it is defined statically. The parameters of VLs affect worst case E2E

delay in the system. BAG parameter and maximum frame size are among the parameters that characterize the VLs.

As mentioned before, a typical industrial AFDX network consists of more than 1000 VLs and 130 ESs [15]. The possible BAG values are 2^n *ms*, where n can in the interval [0, 7] [15]. Packets produced from ESs are periodic. Scheduling of VLs by ESs is known and offsets are used to deal with it. There are also sporadic flows in the system and frames can be generated at any time.

This thesis work uses the network topologies provided in [15], [16], [29] and [30] as examples. Network topology in [15] is given in Figure 4.8. There are 12 ESs (*e1* to *e12*), 5 switches (*S1* to *S5*) and 32 VLs (*v1* to *v32*). Two VLs, namely, *v19* and *v20* provide sporadic flows and the rest of VLs provide periodic flows. The properties of VLs are given in Table 4-1. BAG value and packet size for each ES are chosen to be the same. An example of periodic sequence of frames is given in Figure 4.9. Figure 4.9 and Table 4-1 shows that generation of frames of VLs are independent from each other. The data in Figure 4.8 and Table 4-1 is sufficient for the tool to generate the necessary state machines required by UPPAAL.

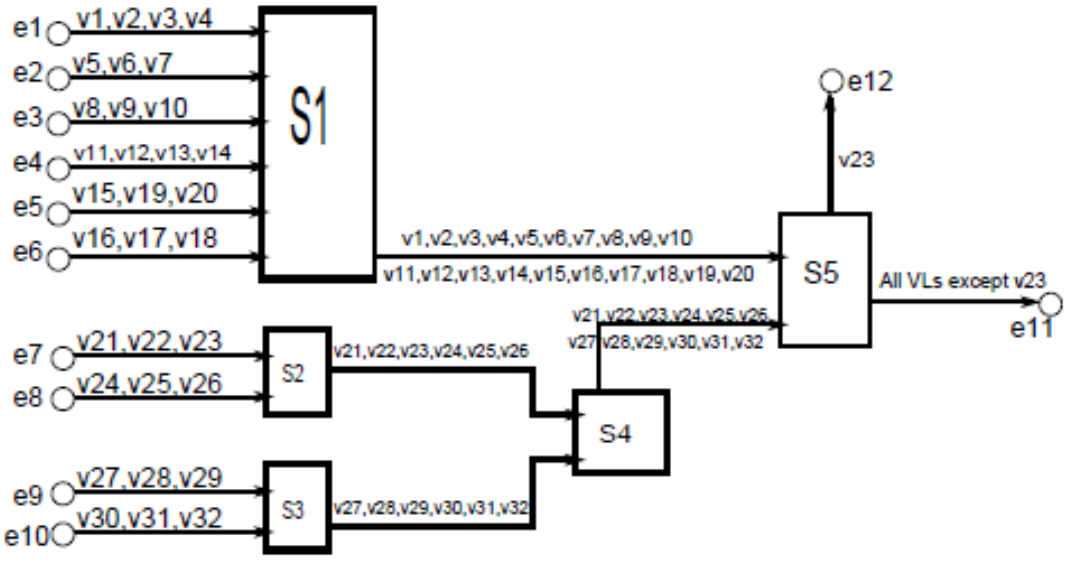

Figure 4.8: Example Network Topology [15]

Table 4-1: Characteristic of VL in the Example Network Topology [15]

VL	BAG (*ms*)	Size (*ms*)	offset (*ms*)	VL	BAG (*ms*)	Size (*ms*)	offset (*ms*)
v1	128	2	0	v17	128	2	32
v2	128	2	32	v18	128	2	64
v3	128	2	64	v19	32	2	n/a
v4	128	2	96	v20	64	2	n/a
v5	128	2	0	v21	128	2	0
v6	128	2	64	v22	128	2	32
v7	128	2	96	v23	128	2	64
v8	128	2	0	v24	128	2	0
v9	128	2	32	v25	128	2	64
v10	128	2	96	v26	128	2	96
v11	128	2	0	v27	128	2	0
v12	128	2	32	v28	128	2	64
v13	128	2	64	v29	128	2	96
v14	128	2	96	v30	128	2	0
v15	128	2	0	v31	128	2	64
v16	128	2	0	v32	128	2	96

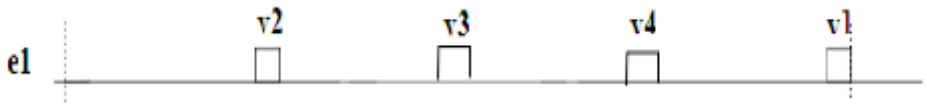

Figure 4.9: Sample Sequence of an End System

4.3 WORST CASE SCENARIO

The delay of a frame, which can be generated periodically or sporadically in AFDX network, is the summation of time spent on each link, waiting times in output buffers of switches and switching delays. When waiting times in output buffers are maximized, the worst case delay is obtained. Therefore, a scenario that maximizes this waiting time should be found. The generation time of frames of each ES is not known since ESs in AFDX networks behave independently from each other. However, the generation time of frames have an impact on worst case E2E delay since it affects the arrival time of output port of switches. Hence, all possible

generation instants should be taken into consideration to determine the upper bound value. All possible scenarios are supported in the tool provided in this thesis work.

In [5], the search space used for the computation of worst case E2E delay is minimized. According to [5], the scenario that all ESs send a frame to their connected switch at the same time with the frame under study is a competitor for worst case E2E delay. This is valid for periodic flows. In Figure 4.10, an example scenario is depicted. All ESs are not shown to prevent confusion. Worst case E2E delay is calculated for VL *v1*. The frame generated from *v1* is queued at the output port of switch *S1* with the frames of *e2* and *e3* at the same time. The sequence is always finished with a frame of *v1* produced by *e1*. This is the frame under study. The sequence is finished with a frame of *v5*, *v6* or *v7* produced by *e2*. Similarly, the sequence is finished with a frame of *v8, v9* or *v10* produced by *e3*. In short, every possible sequence is built in such a way that frame of *v1* coincides with other frames produced by the other neighboring end systems. For the given example topology, this comes up with 9 scenarios 3 of which are given in Figure 4.10.

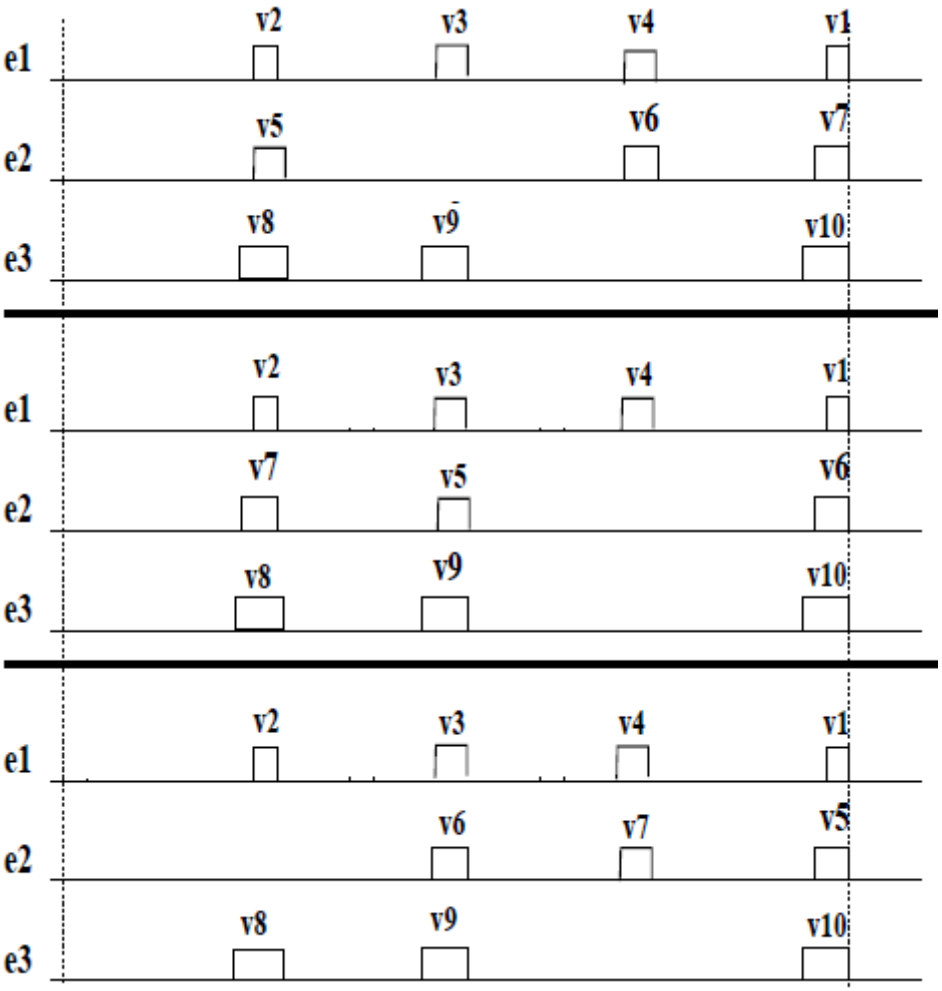

Figure 4.10: Three of Scenarios Used for Worst Case Delay Calculations in Example Network

4.4 MODELING THE SYSTEM WITH FIFO SCHEDULING ALGORITHM

Automata required for the modelling of a given AFDX network architecture can be classified into five different groups as follows [15]:

- **Group G1:** This group consists of VLs belonging to the end system to which VL under study belongs. The data about G1 is input using the first

window of the windows application tool. G1 corresponds to *e1* in paper [15].

- **Group G2:** This group is composed of ESs where they are linked to the same switch as G1. The data related to G2 is input using the second and fourth windows of the windows application tool. There are five ESs and two sporadic VLs are in this group for the example network.
- **Group G3:** This group is composed of VLs connected to other switches. Data related to G3 is input using the third window of the windows application tool. This group consists of four ESs when the example network is considered.
- **Group G4:** G4 is composed of switches in the network. *S1*, *S2*, *S3*, *S4* and *S5* take place in G4 for the example network. All windows store some data related to G4 in our windows application tool.
- **Group G5:** This group of automata is created to measure the E2E delay. Automata of this group are constructed using the final window of the windows application tool.

The buffers at the output ports of switches are constructed as FIFO buffers and they are modeled by *Arrays*. Each element in the array represents a frame. It waits in the queue of switch. Transmission duration shows the packet size of a frame. Nevertheless, this modeling is not suitable for UPPAAL. This is because the size of arrays affects the simulation. Therefore, each element of the array contains only the transmission duration of a frame. Nonetheless, this brings out a problem that the frame cannot be identified, especially the one under study. So as to figure out this problem the frame whose worst case delay value wants to be calculated is modeled by integers (for example *sw1v1* and *sw2v1*). *sw1v1* represents a frame that is transmitted from ES *e1* to switch *S1*. *sw2v1* corresponds to a frame that is sent from switch *S1* to switch *S5*.

So as to catch the worst case E2E delay, a synchronization mechanism is needed between each timed automata. The main idea is to synchronize each automaton in

the system. This synchronization mechanism should be done in UPPAAL after importing the constructed automata into this tool. There are multiple options to do the synchronization. Table 4-2 shows one possible synchronization mechanism for the example network used (the symbol "*" shows the End Systems of group G2). It is explained clearly in [15].

Table 4-2: Synchronization [15]

t	G1	G2	G3		G4				
	e1	(*)	e7,e8	e9,e10	S1	S2	S3	S4	S5
t_0	pause till pause4!	pause till begin!	pause till pause3!	Emit 1st Packet for S3	ready for packet	ready for packet	ready for packet	ready for packet	ready for packet
t_1							ready to emit packet for S4. broadcast pause3!		
t_1			start with pause3!. emit 1st Packet for S2	pause till go3!			pause till go3!		
t_2						ready to emit packet for S4. broadcast go3!			
t_2				resume with go3!			resume with go3!		
t_3								ready to emit packet for S5. broadcast pause4!	
t_3	start with pause4!. broadcast begin!		pause till go4!	pause till go4!		pause till go4!	pause till go4!	pause till go4!	
t_3	emit 1st packet for S1	start with begin!. Emit 1st packet for S1							
t_4					ready to emit packet for S5.broadcast go4!				
t_4			resume with go4!	resume with go4!		resume with go4!	resume with go4!	resume with go4!	
t_5	Stop after 2 Hyper period. broadcast stop!								
t_5		stop with stop!	stop with stop!	stop with stop!					

After synchronizing the automata as described above, the system is ready to measure the worst case E2E delay. So as to perform this step, *Verifier* tab should be selected and the written query *A[]((measure.measure imply clk<value))* should be checked. The formula give result as *true* when there is no scenario leading to *clk* > *value*. Therefore, worst case E2E delay is the maximum value that gives *true* as a result. This value is found by trial and error for the time being.

The constructed automata by the provided tool are given in following figures:

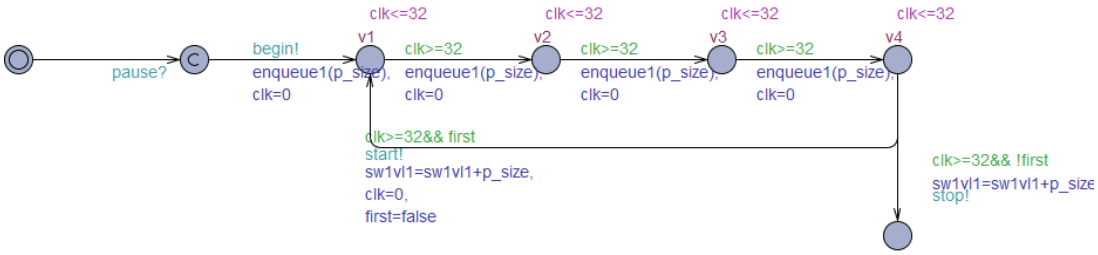

Figure 4.11: Automaton of e1

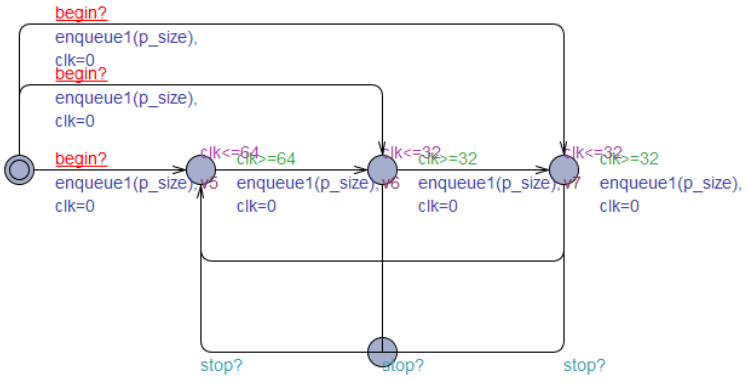

Figure 4.12: Automaton of e2 which is similar to e3, e4, e5, e6

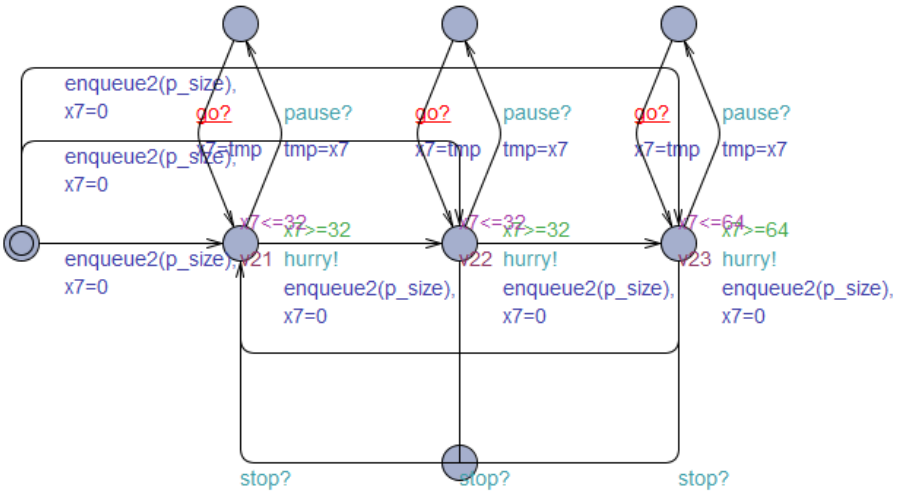

Figure 4.13: Automaton of e7 which is similar to e8, e9, e10

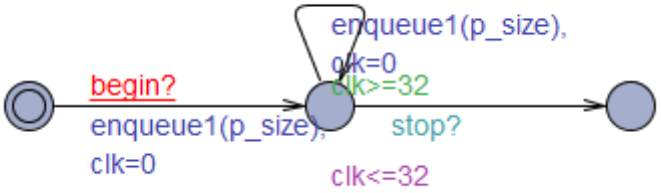

Figure 4.14: Automaton of Sporadic VLs

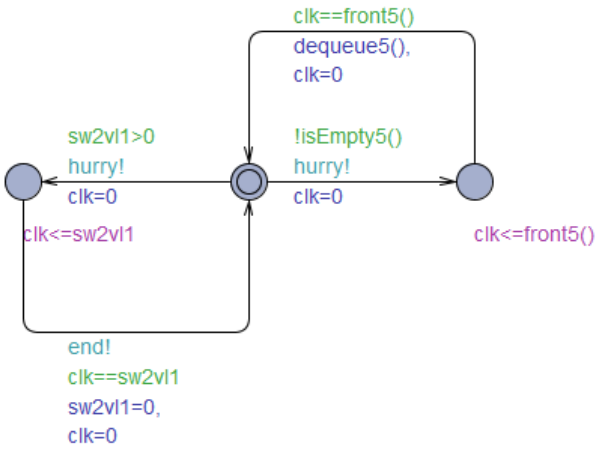

Figure 4.15: Automaton of SW5

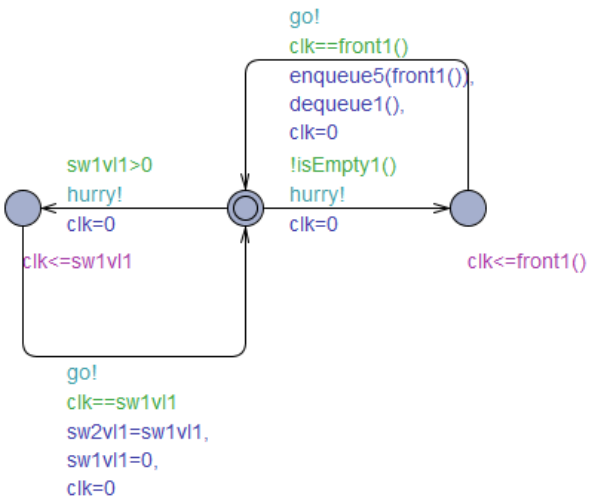

Figure 4.16: Automaton of SW1

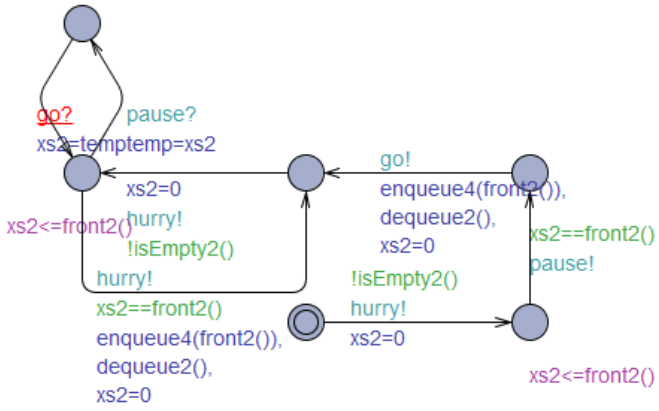

Figure 4.17: Automaton of SW2 which is similar to SW3 and SW4

4.5 MODELING THE SYSTEM WITH ROUND-ROBIN SCHEDULING ALGORITHM

In this scheduling algorithm, groups G1 to G4 are slightly changed. As was mentioned in section 3.2.2, each ES in the network should have its own queue. Therefore, a queue is added to timed-automaton of each ES. This operation should also be applied to switches since RR algorithm must be applied between switch pairs *S1-S4* and *S2-S3* on switches S5 and S4, respectively. However, queues of all

switches described in FIFO algorithm remain same. End System *e1* is used as an example to illustrate the required modification when RR algorithm is to be used.

Unlike FIFO algorithm, a new automaton should be included in system to send packets in a circular order. In Figure 4.31-33 these automata are represented. In Figure 4.18 transmission process in circular order between ESs *e1* to *e6* is shown. Furthermore, the frame under study is considered in this automaton. If frame under study is produced by ES *e1* and is ready to be transmitted, then it comes out from the queue. Similarly, in Figures 4.19 and 4.20 RR algorithm is applied between ES pairs e7-e8 and e9-e10. Counters are used for RR scheduling algorithm. They are represented in Figure 4.20.

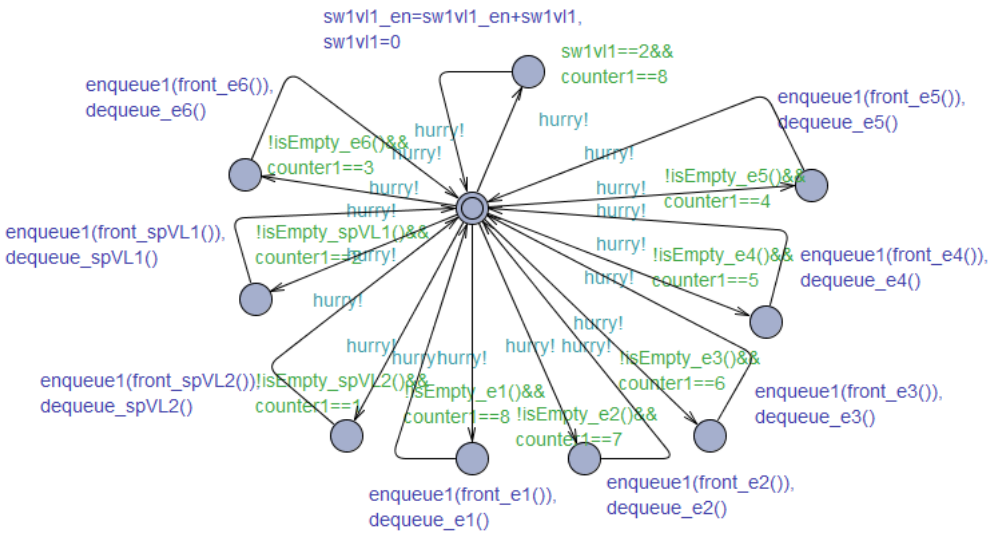

Figure 4.18 Transmission Process between ESs *e1* to *e6*

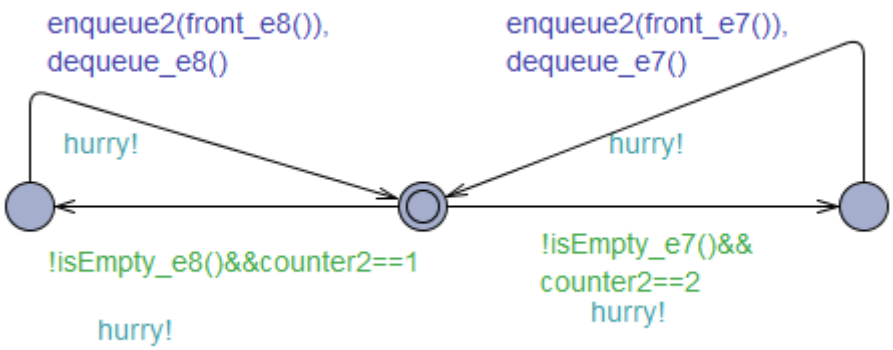

Figure 4.19 Transmission Process between ESs *e7*, *e8* similar to *e9*, *e10*

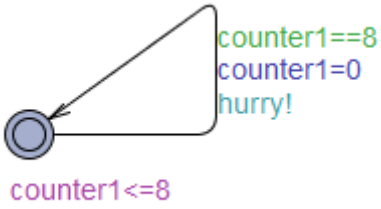

Figure 4.20 Reset Counter

CHAPTER 5

EVALUATION AND IMPLEMENTATION RESULTS

In this chapter, the worst case E2E delay calculated by using FIFO and RR scheduling algorithms is presented and the comparison between these scheduling algorithms is made. Moreover, comparison for worst case E2E delay obtained from different methods is done. For this purpose, the outputs of the developed tool are utilized.

5.1 VERIFICATION IN UPPAAL

The *Verifier* tab in UPPAAL toolbox is used to obtain worst case E2E delay. In this tab of toolbox, users can insert, remove or toggle several properties. With these properties users can check specific actions in the system. For instance, a user can indicate whether a trace is required or not in a specific automaton while the system is running.

So as to calculate the worst case E2E delay, the specific computation tree logic (CTL) formula *A[] ((TemplateName.StateName imply clk<value))* is written. When there is no scenario leading to clk>value, this formula returns true. Therefore, the maximum value of clk gives the worst case E2E delay. The user should input the value of value manually and perform a trial and error procedure in order to find this value.

5.2 WORST CASE END-TO-END DELAY IN FIFO SCHEDULING ALGORITHM

The simulation of the example provided in [15] is completed in approximately 15 minutes on a PC with 2.4 GHz Intel Core processor using 4 GB RAM. The worst case

delay for VL *v1* is 26ms as shown in Figure 5.1. This is shown in Figure 5.1. For the example provided in [16], the worst case E2E delay for VL *vl4* is found to be 16ms.

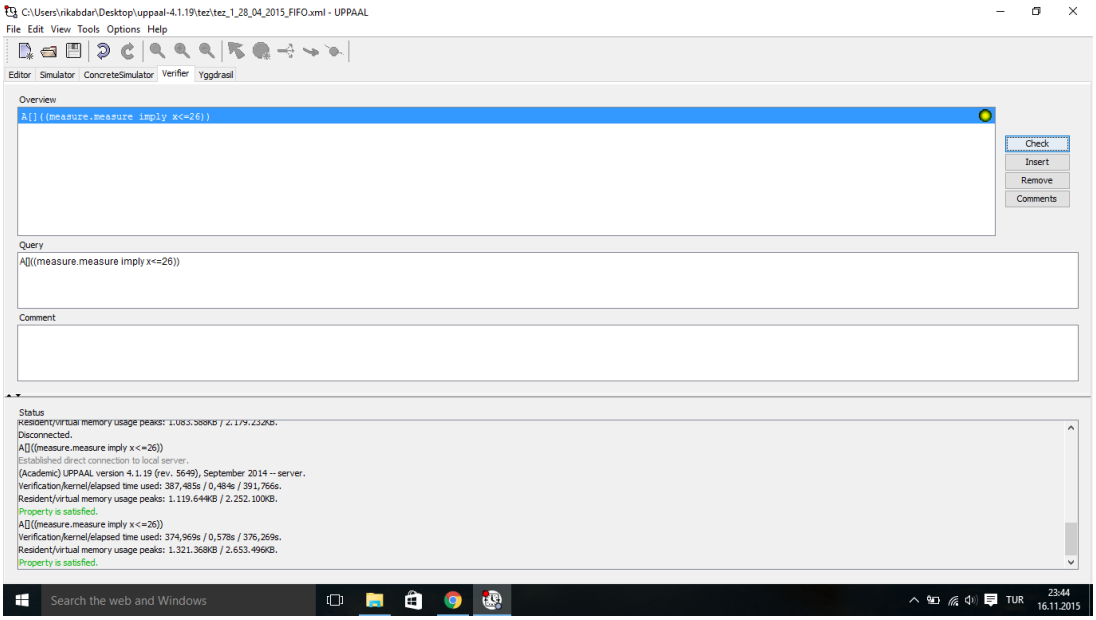

Figure 5.1 Query Given in the Verifier Tab

5.3 WORST CASE END-TO-END DELAY IN RR SCHEDULING ALGORITHM

The simulation of the example provided in [15] is completed in approximately 20 minutes on a PC that has 4 GB RAM. 22ms is found for the worst case delay for VL *v1*.

5.4 PERFORMANCE COMPARISON

A scheduler carries out the scheduling activity. It may aim several purposes such as maximizing throughput and minimizing response time. It may also ensure that the deadline is met in the system.

We have employed the two simplest scheduling algorithms, namely FIFO and RR in our example AFDX topologies and calculated the worst case E2E delay for a

selected VL. These two approaches resulted in two different worst case delay values as expected. In one case, worst case delay with FIFO is calculated to be 26ms, whereas with RR it is computed as 22ms. A time difference between the two approaches is observed. In FIFO scheduling algorithm packets are served in the same order as they arrive. However, in RR scheduling algorithm queues are serviced in a circular order and empty queues are skipped over.

Network topologies in [15], [16], [29] and [30] are taken as samples to compare the methods for calculation of worst case E2E delay. The results are presented in Table 5-1. First column shows where the network topology is taken from. Second column presents the VL which is under study. In other words, the worts case E2E delay of this VL is calculated. The other columns show the methods that are utilized for computing the delay value. NC stands for Network Calculus, MP is abbreviation of Mathematical Programming and last column shows results obtained from tool that is presented in this paper. For the first ten lines, the MP method is compared with the tool. It is seen that tool gives approximately 8% better result. However, MP method calculates worst case E2E delay using IBM CPLEX in order of miliseconds in [29] but our tool gives results in order of seconds. Lines 11-13 in this table represent comparison of three methods. In these lines NC and MP methods give overestimated result. For the last two lines, network topology given in [15] and [16] are used and Trajectory approach is analyzed with our tool. Like other methods Trajectory approach gives more drastic result than our tool's result. As seen from the table, network modeling approach gives more improved results. Therefore, this approach is preferred.

Table 5-1: Worst Case E2E Delay Results with Different Methods

Paper Ref. Num.	VL US (*)	NC Method	MP Method	Trajectory Method	Tool Support
29	v1	-	580 µs	-	520 µs
29	v2	-	360 µs	-	320 µs
29	v3	-	580 µs	-	520 µs
29	v4	-	320 µs	-	280 µs
29	v6-e7 (**)	-	920 µs	-	880 µs
29	v6-e8 (**)	-	1280 µs	-	1220 µs
29	v7	-	1240 µs	-	1160 µs
29	v8	-	920 µs	-	880 µs
29	v9	-	880 µs	-	820 µs
29	vx	-	680 µs	-	620 µs
30	v1-e7 (**)	711,07 µs	692 µs	-	676 µs
30	v2–e7 (**)	711,07 µs	672 µs	-	656 µs
30	v5-e8 (**)	376,54 µs	372 µs	-	352 µs
15	v1	-	-	28 ms	26 ms
16	v4	-	-	18 ms	16 ms

(*): It defines the VL under study.

(**): Is shows the VL with destination ES.

CHAPTER 6

CONCLUSION AND FUTURE WORK

AFDX is defined to meet the requirements of avionics systems. It consists of three basic elements and the main task in this network is to transmit packets within a specified time.

AFDX is required to meet QoS properties and its real-time performance is the most important issue. Therefore, a worst case E2E delay analysis should be computed on an AFDX design. There exist five methods to do this. Network calculus computes the worst case E2E delay analytically. Trajectory approach is another one and delay value is determined by using the information about the trajectory of a packet in this approach. Mathematical rules are used in the third approach. The network characteristics are reflected more clearly in the simulation based analysis. The last approach is network modeling. Modeling approach is used in this thesis work since other approaches have vaious drawbacks when they are compared.

Model checking of AFDX networks employs timed automata and UPPAAL is utilized to investigate the worst case E2E delay analysis. In order to provide the state machines reuired in UPPAAL, a tool which employs a windows form application is designed. In this application, all data related to the given network topology is input from the user. The output of the application is templates that represent all ESs and switches and that are ready to be imported into UPPAAL directly. The application is designed to generate state machines by considering simple FIFO scheduling algorithm at the output ports of switches. However, a second alternative, that is the RR approach, can also be used and E2E worst case delay can be computed.

Worst case E2E delay is found by using an appropriate CTL formula in the *Verifier* section of UPPAAL. RR scheduling algorithm is applied to the system afterwards. Slight changes are needed while applying RR. The worst case E2E delay of the same ES is computed. It is observed that RR scheduling algorithm gives a better result in comparison to FIFO approach for the VL under study.

Network topologies given in reference papers are taken as sample to compare methods mentioned in section 2 with tool designed in this thesis work. The obtained results from our tool are analysed with the ones represented in reference papers. It is seen that network modeling approach gives more improved result.

This thesis work can be extended as follows: The packet size is fixed in this work but it may be variable. Synchronization signals are modified according to user's taste, which can be fixed as the future work. Finally, the worst case E2E delay is calculated using a trial and error approach since the upper bound value in the CTL query is provided manually for the time being. This can also be automatized and the maximum value can be found accordingly.

REFERENCES

[1] Boudec J.-Y. Le, Thiran P., "Network Calculus: A Theory of Deterministic Queuing System for the Internet", LCNS 2050, pp. 3-26, 2012.

[2] Charara H., Scharbarg J.-Y., Ermont J., Fraboul C., "Methods for bounding end-to-end delays on an AFDX network", 18th Euromicro Conference on Real-Time Systems, 10 pp. -202, 2006.

[3] R. Nishi, G. Zhu, Y. Savaria, "Optimal Scheduling Policy for AFDX End-Systmes with Virtual Links of Identical Bandwidth Allocation Gap Size", 25th IEEE Canadian Conference on Electrical and Computer Engineering, pp. 1-4, 2012.

[4] Liu C., Wang T., Zhao C., "Worst-case flow model of VL for worst-case delay analysis of AFDX", Electronics Letters Vol. 48 No.6, 2012.

[5] Adnan M., Jean-Luc S., Fraboul C., "Minimizing the search space for computing exact worst-case delays of AFDX periodic flows", In Proc. of the 6th SIES, Vasteras, June 2011.

[6] Scharbarg J.-Y., Ridouard F., Fraboul C., "A Probabilistic Analysis of End-To-End Delays on an AFDX Avionic Network", IEEE Trans. on Industrial Informatics, Vol.5 No.1, 2009.

[7] Lina D., Dong S., Xingxing Z., Q. Hu, "The research of AFDX system simulation model", Multimedia Technology (ICMT), pp. 1-4, 2010.

[8] Dong S., Xingxing Z., Lina D., Qiong H., "The Design and Implementation of the AFDX Network Simulation System", Multimedia Technology (ICMT), pp. 1-4, 2010.

[9] Chen X., Xiang X., Wan J., "A Software Implementation of AFDX End System", Int. Conf. on New Trends in Information and Service Science, pp. 558-563, 2009.

[10] Sambou B., Peyrard F., Fraboul C., “Scheduling Avionics Flows on an IEEE 802.11e HCCA and AFDX Hybrid Network”, IEEE Symposium on Computers and Communications (ISCC), pp. 205-212, 2011.

[11] Yao M., Qui Z., Kwak K., “Leaky Bucket Algorithms in AFDX”, Electronics Letter Vol. 45 No.11, pp. 543-545, 2009.

[12] Bisson K., Troshynski T., “Switched Ethernet Testing for Avionics Applications”, IEEE A&E Systems Magazine, pp. 31-35, May 2004.

[13] Sheikh A. A., Brun O., Cheramy M., Hladik P.-E., “Optimal design of virtual links in AFDX networks”, Real-Time Syst, pp. 308-336, 2013.

[14] “http://www.aim-online.com” [Online], AIM GmbH Inc., March 2010, Available: http:// www.aim-online.com [Accessed 06 December 2014].

[15] Adnan M., Scharbarg J.-L., Ermont J., Fraboul C., “An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows”, IEEE, ETFA, pp. 1-8, 2012.

[16] Adnan M., Scharbarg J.-L., Ermont J., Fraboul C., “An improved timed automata model for computing exact worst-case delays of AFDX periodic flows”, IEEE ETFA, pp. 1-4, 2011.

[17] Bauer H., Scharbarg J.-L., Fraboul C., “Applying and optimizing Trajectory approach for performance evaluation of AFDX networks”, Real-Time and (Networked) Embedded Systems - 14st ETFA, pp. 1-8, 2009.

[18] Alur R., “Timed Automata”, Technical Reports (CIS), pp. 1-28, January 1998.

[19] Alur R., Dill D. L., “Fundemental Study A theory of timed automata”, Theoretical Computer Science Elsevier, pp. 183-235, 1994.

[20] Behrmann G., Daivd A., Larsen K. G., “A Tutorial on UPPAAL 4.0”, Department of Computer Science Aalborg University Denmark, 2006.

[21]Vaandrager F., “A First Introduction to UPPAAL”, In J. Tretmans, editor. Quasimodo Handbook, 2014.

[22] Behrmann G., “Introduction to UPPAAL”, Aalborg Universtiy, April 13 2015.

[23] David A., Pettersson P., “UPPAAL Tutorial Introduction”, IEEE Real-Time Systems Symposium (RTSS), pp. 1-36, 2005.

[24] Huang X., Singh A, Smolka S. A., “Using Integer Clocks to Verify the Timing-Sync Sensor Network Protocol”, Proceedings of NFM, Washington D. C. USA, pp. 77-86, 2010.

[25] Vörös A., “Short UPPAAL Introduction”, Budapest University of Technology and Economics Department of Measuremetn and Information Systems Fault Tolerant Systems Research Group, 13 October 2013.

[26] AFDX Protocol Tutorial, CONDOR Engineering, 2005.

[27] Bauer H., Scharbarg J.-L., Fraboul C., “Applying Trajectory approach to AFDX avionics network”, Proc. 14th Int. Conf. Emerging Technol. Factory Autom., pp.1 - 8, 2009.

[28] Sheikh A. A., Brun O., Chéramy M., Hladik P-E., “Optimal design of virtual links in AFDX networks”, Real-Time Syst, pp. 308-336, 2013.

[29] Malta L., Oliveira R. D. S., “A Model to Calculate Exact End-to-End Delay of Sporadic Flows On AFDX Network Using Mathematical Programming”, SBESC Brazilian Symposium, pp. 87-92, 2012

[30] Li X., Scharbarg J.-L., Fraboul C., “Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis”, ETFA IEEE Conference, pp. 1-8, 2010.

[31] Lv T., Hu N., Wu Z., Huang N., “The Analysis of End-to-End Delays Based on AFDX Configuration”, ICRMS International Conference, pp. 1296-1300, 2011.

[32] Tawk M., Zhu G., Savaria Y., Liu X., Li J., Hu F., “A Tight End-to-End Delay Bound And Scheduling Optimization of An Avionics AFDX Network”, DASC, pp. 7B3-1 7B3-10, 2011.

[33] Kemayo G., Ridouard F., Bauer H., Richard P., “Optimistic problems in the trajectory approach in FIFO context”, ETFA IEEE 18th Conference, pp. 1-8, 2013.

[34] Bauer H., Scharbarg J.-L., Fraboul C., “Applying and optimizing Trajectory approach for performance evaluation of AFDX avionics network”, ETFA IEEE Conference, pp. 1-8, 2009.