# Response time analysis in AFDX networks

J. Javier Gutiérrez, J. Carlos Palencia, and Michael González Harbour

*Computers and Real-Time Group, Universidad de Cantabria, 39005-Santander, SPAIN*
*{gutierjj, palencij, mgh}@unican.es*

## Abstract[1]

The ARINC-664, Part 7 (AFDX) standard defines a communications network based on Ethernet and the UDP/IP protocols. Contrary to general-purpose Ethernet, the timing behavior in AFDX is deterministic due to the use of special network switches and end-systems with static routing tables and traffic regulation at the sending end through mechanisms called virtual links. Even though the latencies in this network are bounded, there are scheduling and contention effects that need to be analyzed. In this paper we develop a response-time analysis of the network including the scheduling of the virtual links and contention in the end-systems and in the switches. This response time allows us to obtain worst-case latencies and output jitter for the network messages. These results can be integrated with the response time analysis in other resources to obtain end-to-end response times in complex distributed systems.

## 1. Introduction

AFDX (Avionics Full Duplex Switched Ethernet) is a communications network defined in the ARINC-644, Part 7 standard [1] and based on the use of point to point full duplex ethernet links and special purpose switches in which the routing of messages is preconfigured so that there is no delay in the discovery of routing addresses through network protocols that could interfere with the transmission of application messages. In addition, AFDX provides two redundant hardware communication links for fault-tolerant operation.

AFDX [3][10] defines the communication process among end-systems (processing nodes) where bandwidth and bounded latency are guaranteed, although the particular jitter for a flow of communication packets between two end systems is not fixed, since it depends on the global network traffic at a given time.

1. This work has been funded in part by the Spanish Ministry of Science and Technology under grant number TIN2008-06766-C03-03 (RT-MODEL), and by the European Commission under project FP7/NoE/214373 (ArtistDesign).

This paper describes methods for performing response-time analysis (RTA) in AFDX deterministic switched networks. The challenges are modelling the queuing effects in the end-systems and in the AFDX switches, and modelling the end-to-end response times of the communications, including the case with multicasting.

We have defined a real-time model for a communications network based on AFDX. From that model, we have developed a response time analysis for AFDX networks that can be integrated with the other response-time scheduling analysis. In this way we have developed end-to-end response time analysis techniques for complex distributed systems.

The paper is organized as follows. In Section 2 we describe the AFDX network from the point of view of its scheduling properties and timing behavior. Section 3 states the assumptions for the analysis, while Section 4 describes the model of the AFDX network. Section 5 derives the response time analysis, and Section 6 describes how to combine this analysis with response times in other resources to obtain an end-to-end RTA. Section 7 shows a simple example to illustrate the application of the techniques developed in previous sections. Finally, Section 8 presents a summary of the results, the conclusions, and future work.

## 2. The AFDX Network

Normal communications for AFDX is made interchanging messages through communication ports. The communication ports are defined in the ARINC 653 standard [2], which defines the interface of a partition-based operating system for use in avionics systems. There are two different types of ports: sampling or queueing. Queueing ports are required to manage at least 8Kbytes of data.

For transmission there is no difference in the behavior of both types. Messages that are generated are directly queued in the transmission buffer. For message reception, the behavior of these ports is different:

- *Sampling Port*: the arriving message overwrites the current message stored in the buffer.

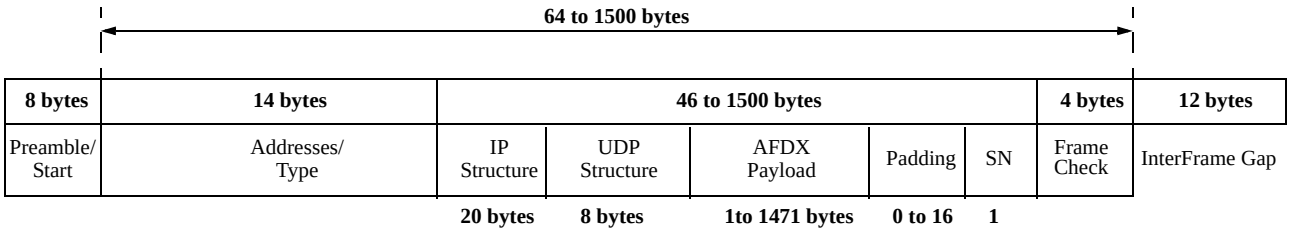


Figure 1. Scheme of the Ethernet frame

- *Queueing Port*: the arriving message is appended to a FIFO queue.

Another mode of transfer in avionics services is the Trivial File Transfer Protocol (TFTP) and communication with compliant networks via SAP (Service Access Point) ports. However, in this paper we have focused on the normal communication mechanism through sampling or queueing ports.

The ARINC 653 API has operations to send or receive messages to or from these AFDX communication ports. The messages are driven through the transmission protocol stack based on the UDP/IP protocol, and they might be fragmented into packets according to the traffic regulation parameters. The packets are sent through two redundant networks; the redundancy management of the packets sent or received is made by specific Ethernet hardware.

The traffic regulation is made via *Virtual Links* defined (see 1.2.1 “Virtual Link” in attachment 1 to [1]) as conceptual communication objects to establish a logical unidirectional connection from one source end system to one or more destination end systems having a dedicated maximum bandwidth. Each virtual link (VL) is characterized by two parameters used for traffic regulation:

- The largest Ethernet frame (*Lmax*), which is a value in bytes.
- The Bandwidth Allocation Gap (*BAG*), which is a power of 2 value in the range [1,128]. The *BAG* represents the minimum interval in milliseconds between Ethernet frames transmitted on the VL.

Each virtual link has a FIFO queue for all the fragmented packets to be transmitted on this VL with its appropriate bandwidth. Several AFDX communication ports may share the same VL to transmit their packets. Furthermore, in a partitioned system using ARINC 653, several partitions or tasks in the same or in different partitions could share the same AFDX port. Sharing ports or VLs causes a poor schedulability of the system, as there is no way to prioritize messages and they will be enqueued in FIFO order. The VL queue for packets is a source of contention for the messages transmitted on an AFDX network.

In order to avoid the fragmentation of messages for sampling ports, it is recommended to adjust the *Lmax* of the virtual link to accommodate the complete message. On the other hand, queuing ports can support messages of different sizes up to a maximum of 8 Kbytes, so fragmentation may be needed. Fragmentation may also be needed when very long packets could excessively delay the transmission of other contending messages with very short deadlines.

The Virtual Link Scheduler is in charge of selecting the next packet to be transmitted according to the allocated bandwidth for each VL. This scheduler selects the first packet from a VL queue that is ready to be transmitted. When several VLs are ready to transmit then they are selected in turns until all of their messages have been transmitted. This choice introduces jitter for the transmission over any of the VLs, which is bounded by a maximum value defined in the specification (subclause 1.2.4.3 “Jitter” in attachment 1 to [1]).

The maximum allowed jitter on each VL at the output of the end system should comply with both of the following formulas:

$$MaxJitter \leq 40\mu s + \frac{\sum_{i \in \{set\ of\ VLs\}} ((20 + Lmax_i) \cdot 8)}{N_{bw}} \qquad (1)$$

$$MaxJitter \leq 500\mu s$$

where, $N_{bw}$ is the speed of the Ethernet link in bits per second, 40 μs is the typical minimum fixed technological jitter, and the maximum jitter is expressed in microseconds. The value 20 is the number of bytes to be added to each Ethernet frame (see Figure 1): 8 bytes for the preamble and the start of frame delimiter (SFD) and 12 bytes for the inter-frame gap (IFG).

We also have to take into account that the minimum Ethernet frame has 64 more bytes, which can be used to send AFDX payloads between 1 and 17 bytes. This means that at least 84 bytes are always transmitted. The maximum Ethernet frame has 1518 bytes for an AFDX payload up to 1471 bytes. So 1538 is the maximum number of bytes

transmitted per packet. The total amount of bytes sent through the network can be obtained in terms of the AFDX payload according to the scheme of the Ethernet frame with UDP/IP illustrated in Figure 1.

A virtual link can be composed of a number of Sub-Virtual Links, all of them having a dedicated FIFO queue which is read on a round robin basis by the main VL FIFO queue. The round robin algorithm works over IP fragmented packets.

The ARINC 664 specification [1] describes that it is the system integrator's responsibility to determine that for the chosen end system configuration and implementation the 500 μs limit in Eq. (1) is not exceeded. This specification also defines the following two limiting cases to mathematically treat the allowed latency in the end system transmission (subclause 1.2.4.3 "Jitter" in attachment 1 to [1]):

- For those messages that are shorter than *Lmax* (and therefore do not require fragmentation) and that are produced at a frequency that is equal to or lower than the *BAG* of the VL, the total allowed latency is:

$$MaxLatency \leq BAG + MaxJitter + L_T \quad (2)$$

  where, $L_T$ is the technological latency in the transmission, defined as the time required for a packet to go by the end system hardware when there is no contention from other messages. The value of $L_T$ should be bounded and lower than 150μs (see subclause 1.2.4.1 "Latency" in attachment 1 to [1]).
- For those messages requiring fragmentation or that are produced in bursts, there could be *p*-1 packets already waiting to be processed in the VL FIFO queue, and then the latency for packet *p* on the VL can be calculated as follows:

$$MaxLatency(p) \leq p \cdot BAG + MaxJitter + L_T \quad (3)$$

Once a packet is ready to be transmitted, it is sent to the switch using the full capacity of the physical link. The switch delivers the packet from the incoming port (where the source end system is connected) to the outgoing port or ports (where the destination end systems are connected) in a store and forward way. A new contention point, and a new source of jitter appears in the FIFO queue where packets should wait to be sent to the destination end system. The latency introduced when a packet is delivered from the incoming to the outgoing port must also be taken into account (known as the hardware latency of the switch). Then, once the packet is ready to be transmitted from the FIFO queue of the outgoing port, it is sent to the destination end system using the full capacity of the physical link.

At the destination end system the packet is driven through the reception protocol stack. When a message is completely received, it is enqueued at the corresponding AFDX port, which could potentially overflow. Similar to the $L_T$ value, the technological latency of the end system in reception, $L_R$,should be bounded and lower than 150μs (see subclause 1.2.4.1 "Latency" in attachment 1 to [1]).

## 3. Assumptions for the analysis

According to the ARINC 664 specification [1] and in order to establish a basic model, we have made the following assumptions:

1. Applications exchange data on the network exclusively using AFDX communication ports, with messages sent from one source port in an end system to just one destination port in another end system. We will later eliminate this restriction and allow the use of multicast messages.
2. The communication process starts in an end system when a message is sent to an AFDX communication port using the AFDX API, and finishes when the message has reached the destination port and has been received using the corresponding operation of the AFDX API.
3. Only VLs (virtual links) are considered; sub virtual links are not considered for the moment.
4. All the queues are FIFO, so no priorities have been considered.
5. Initially we consider that messages only cross one switch, but we will later extend the analysis to the use of multiple switches.
6. The model of the latency of the hardware in the switch to manage a packet from the incoming port to the outgoing port should be provided by the manufacturer. We will assume a simple model based on a bounded latency, for the purpose of developing an initial analysis.
7. The latency of the end system hardware for transmission or reception is also provided by the manufacturer or can be calculated somehow. We will assume a simple model based on a bounded latency.
8. The latency in the physical link due to the propagation of a bit is insignificant compared to the rest of the latencies, assuming short distance communications, and therefore we will not take it into account. In the same

way as it is calculated in [10], the latency for bits transmitted through a fiber optic link of 100 meters length is around 0.5 μs. The transmission times for the minimum and the maximum frame sizes (84 and 1538 bytes) at 100 Mbps are 6.72 μs and 123.04 μs respectively.

9. We will consider variable-size messages for each message sequence, within a minimum and a maximum size.

10. We assume that the queues in the switches and in the end systems are large enough to accommodate the worst-case traffic.

## 4. Modelling

This section is devoted to describing the model of the communication process across an AFDX network, including the traffic produced by the applications and the identification of the different stages involved in the communication process. The objective is to allow the calculation of the worst-case and the best-case latencies or transmission times for any message from the instant when the message is sent to the AFDX port by an application task (called the *release time*) until the message is received by a destination task from the corresponding AFDX port (called the *arrival time*). The resulting worst- and best-case latencies can be used to calculate offset and jitter terms for the overall analysis of the distributed system as described in Section 6.

The task model used for the analysis of the AFDX network is concentrated on the elements involved in the communication. In this simplified model we assume that applications are composed of tasks released by a stream of periodic events. These tasks execute one instance, or *job*, per event received, and therefore each task executes an infinite stream of jobs, one for each period or event activating it. Each of these tasks can send messages to the AFDX ports at specific times and rates.

We will distinguish two types of messages:

- *Synchronized*. The release times of these messages are relative to a general and common reference of time. This time reference might be for example the start of the MAF (major frame) for ARINC 653 systems. An offset is defined to represent the interval between the start of the MAF and the earliest release of the message.
- *Non-synchronized*. There is no restriction on the temporal relation between message releases. These messages can be released at any time.

A message stream $\sigma_i$ in AFDX is characterized by the following parameters:

- Worst-case number of bytes of a message ($M_i$): it is the maximum number of bytes of the message payload. It allows us to obtain the number of packets $p_i$ into which the message is fragmented, for the worst-case analysis.
- Worst-case number of bytes of a packet payload ($Np_i$): it is the maximum number of bytes of the payload of a single packet.
- Total worst-case number of bytes of a packet ($N_i$): it is the maximum number of bytes of a single packet, including the message payload ($Np_i$) and the overhead bytes.
- Best-case number of bytes of a message ($M_i^b$): it is the minimum number of bytes of the message payload. It allows us to obtain the number of packets $p_i^b$ into which the message is fragmented, for the best-case analysis.
- Best-case number of bytes of a packet payload ($Np_i^b$): it is the minimum number of bytes of the payload of a single packet.
- Total best-case number of bytes of a packet ($N_i^b$): it is the minimum number of bytes of a single packet, including the message payload ($Np_i^b$) and the overhead bytes.
- Period ($T_i$): it is the minimum time between the release of two messages of the $\sigma_i$ stream for non-synchronized messages and the regular interval of time at which the messages are released for synchronous messages.
- Offset ($\Phi_i$): in synchronized messages, it is the time relative to the start of the MAF or common time reference that the release of the message will be delayed. Currently we will not use this value, but in future work we plan to develop analysis techniques that would use this value to reduce the pessimism of the analysis for systems with synchronized messages.
- Release jitter ($J_i$): it is the time expressing the variability in the release of the messages with respect to the period. It usually depends on the output jitter of the task sending the message.
- Source (*Source*($\sigma_i$)): it identifies the AFDX port and end system from which the message is sent.
- Worst-case latency ($L_i$) and best-case latency ($L_i^b$): these are the results of the analysis, and they represent respectively the worst and the best latencies measured since the message is released by enqueueing it at the AFDX sending port until it arrives at the AFDX destination port.

Virtual links are in charge of regulating the transmissions from each end system. Virtual link $VL_j$ is characterized by the following information:

- $BAG_j$ (Bandwidth Allocation Gap): it is the minimum interval in milliseconds between Ethernet frames

transmitted on $VL_j$. Valid values are $2^n$ with $0 \le n \le 7$ .

- $Lmax_j$ (Largest Ethernet frame): it is the number of bytes that can be transmitted on $VL_j$ at any $BAG_j$ interval.
- SourcePorts($VL_j$): they are the AFDX port or ports which are routed through $VL_j$. There may be one or more ports but only one end system for a given VL.
- DestinationPorts($VL_j$): they are the AFDX ports where the messages are delivered. There may be one or more ports corresponding to one or more end systems.

For systems where the messages should cross more than one switch, the routing information for each switch has to be preconfigured in order to determine the connection between the input and the output ports. If messages only cross one switch, the routing information can be extracted directly from the VL information.

In any case, there are parameters linked to the hardware which are needed to determine the latency of the complete transmission of a message:

- Speed of the Ethernet link ($N_{bw}$): it is the number of bits per second transmitted through the physical link.
- Technological latency in the AFDX hardware ($L_T$): it is the time taken by the Ethernet hardware to pick up a message from the AFDX port, to enqueue it at the VL queue (splitting it into packets if neccesary), and finally to send the first bit of the Ethernet frame. To calculate this value it is assumed that there are no other messages or packets to transmit. This parameter is defined in the ARINC-644, Part 7 standard (subclauses 1.2.4.1 and 1.2.4.3 in attachment 1 to [1]) and is limited to a maximum of 150 μs, irrespective of whether one or more messages are sent. We decompose it into the sum of two other parameters:
  - Minimum technological latency in the AFDX hardware ($L_{Tmin}$): This is the minimum value of $L_T$.
  - The minimum fixed technological jitter ($J_{Tech}$): this parameter is defined in the ARINC-644, Part 7 standard (first equation and "note" in subclause 1.2.4.3 "Jitter" in attachment 1 to [1]), with a typical value of 40μs. It is the variable part of $L_T$.
- Technological latency in reception ($L_R$): it is the worst-case time taken by the Ethernet hardware since the last bit of an ethernet frame has arrived until the message is enqueued at the AFDX port. The best-case value is called $L_R^b$.
- Switch hardware latency ($L_S$): it is the worst-case time taken by the switch to manage a packet from the incoming port buffer to the outgoing port buffer. The best-case value is called $L_S^b$.

Another three parameters can be defined to model the Ethernet frame and the protocol used:

- The Ethernet overhead ($O_{Eth}$): it is the number of overhead bytes to be added to each Ethernet frame (Preamble, Start Frame Delimiter, and Inter Frame Gap). Its value is 20 bytes.
- Protocol overhead ($O_{Prot}$): it is the number of overhead bytes corresponding to the protocol used for communications. Its value is 47 bytes for the UDP/IP used in AFDX.
- Minimum Ethernet frame ($N_{min}$): it is the number of bytes of the minimum Ethernet frame. Its value is 64 bytes.

In the model for the communication process across an AFDX network we can consider the following stages:

1. Sending operation to reach the FIFO queue of the AFDX port ($C_{Send}$). This is the overhead of the message send operation provided by the API. We need to evaluate it as part of the execution time of the task sending the message.
2. Message delivery through the network. This is the process starting when the message is released from the source AFDX port and finishing when the message is queued at the destination AFDX port. It involves the transmission through the end-system hardware, through the network links and through one or more switches.
3. Receiving operation to get a message from the AFDX receiving port ($C_{Receive}$). In the same way as for the sending operation, we have to evaluate the overhead for this operation provided by the API in order to add this extra execution time to the task receiving the message.

To model the latency of the second stage, message delivery through the network, we divide it into the following latencies:

- *Step 1*. Latency of scheduling the virtual links ($L_{VL}$): it is the time needed to deliver a message from the AFDX port to the physical link. It takes into account the time needed to deliver all the packets if the message has been fragmented and the interference of other messages that can be awaiting in the VL queue.
- *Step 2*. Latency of the transmission to the switch ($L_{Tr}$): it is the time needed to send the last packet of a message to the switch across the physical link. Notice that the time needed to send the previous packets (sent in previous *BAG*s) is already included in $L_{VL}$.
- *Step 3*. Latency of the switch management ($L_{Sw}$): it is the time needed to deliver the last packet of a message from the incoming to the outgoing ports of the switch, plus the interference that the packet can suffer due to other

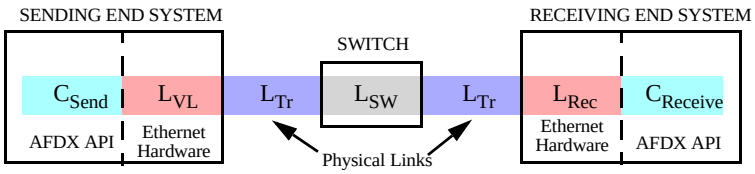


Figure 2. Latencies of the communication process

messages sent to the same destination end system. Notice that the time needed to deliver the previous packets (corresponding to previous *BAG*s) is already included in $L_{VL}$, as the minimum *BAG* is 1ms, while the maximum transmission jitter, according to the ARINC-644, Part 7 standard (first equation and "note" in subclause 1.2.4.3), is 0.5 ms.

- *Step 4*. Latency of the transmission to the destination end system ($L_{Tr}$): it is the time needed to send the last packet of a message from the switch to the end system across the physical link, and is the same as for the transmission to the switch (Step 2).
- *Step 5*. Latency of the message management at the destination end system ($L_{Rec}$): it is the time needed to enqueue the message at the AFDX port.

Figure 2 shows the five steps with their latencies as described above, as well as the send and receive stages. If a packet has to cross more than one switch, steps 2 to 4 need to be replicated for each of the switches that the packet crosses.

## 5. Analysis of AFDX Systems

This section derives schedulability analysis techniques that can be applied to the real-time model for a communications network based on the ARINC 664 Part 7 (AFDX) standard. We first focus on the analysis with just one switch, both for non-synchronized messages as well as for synchronized messages. Then we extend the analysis to the use of multiple switches.

In this section the analysis is based on a worst-case situation for messages exchanged across the network and following the assumptions made for the model. This worst-case situation is pessimistic and could be enhanced with the development of new schedulability analysis techniques taking into account offsets, probably by adapting the existing ones for fixed priorities or EDF [6][8][9].

The analysis in this section is only for non-synchronized ,messages. Synchronized messages can be analyzed pessimistically by treating them as if they were non synchronized. It seems feasible to apply the offset-based response time analysis in a similar way for the calculation of $L_{VLQ(ik)}$, the worst-case latency due to the messages that can be awaiting on $VL_k$. The relative phasings between synchronized messages would be modelled through offsets. This analysis could eliminate part of the pessimism in the treatment of synchronized messages, and we leave it as future work.

### 5.1. Analysis for non-synchronized messages

In this subsection we describe the analysis techniques to calculate the latencies of steps 1 to 5 (see Figure 2) in the communication process for messages produced by non-synchronized tasks with jitter.

#### 5.1.1. Transmission of the last packet to the switch or to the end system, $L_{Tr}$ (steps 2 and 4)

The number of packets of a message belonging to stream $\sigma_i$ being sent through $VL_k$ can be calculated as follows, for the worst case:

$$p_i = \left\lceil \frac{M_i}{Lmax_k - O_{Prot}} \right\rceil \quad (4)$$

where $O_{prot}$ is the protocol overhead in bytes.

The worst-case latency of a packet transmitted through the Ethernet link depends on the speed of the link, $N_{bw}$, and the worst-case number of bytes of the packet $N_i$. The following formula calculates this latency for a packet belonging to the message stream $\sigma_i$:

$$Latency = \frac{N_i \cdot 8}{N_{bw}} \quad (5)$$

where, *Latency* is measured in seconds, $N_{bw}$ in bits per second (bps), and $N_i$ is the worst-case total amount of bytes of the packet:

$$\begin{aligned} N_i &= O_{Eth} + N_{min} && Np_i \in [1,17] \\ N_i &= O_{Eth} + O_{Prot} + Np_i && Np_i \in [18,1471] \end{aligned} \quad (6)$$

where, $Np_i$ is the amount of bytes corresponding to the packet payload.

The size of the last packet of a message belonging to stream $\sigma_i$ being sent through $VL_k$ can be obtained for the

worst case by applying Eq. (6) to the worst payload for this packet, $Np_{i,last}$. This payload can be calculated as follows:

$$Np_{i,\,last} = M_i - (p_i - 1) \cdot (Lmax_k - O_{Prot}) \quad (7)$$

This equation calculates the number of packets by subtracting an integer number of maximum-size payloads from the message size. Using the worst-case payload of the last packet we can calculate its total size using Eq. (6):

$$N_{i,\,last} = O_{Eth} + N_{min} \qquad Np_{i,\,last} \in [1,17] \quad (8)$$
$$N_{i,\,last} = O_{Eth} + O_{Prot} + Np_{i,\,last} \qquad Np_{i,\,last} \in [18,1471]$$

And then we can calculate the worst-case latency of a last packet transmitted through the Ethernet link applying Eq. (5) with this size:

$$L_{Tr(i)} = \frac{N_{i,\,last} \cdot 8}{N_{bw}} \quad (9)$$

The same calculation can be done for the largest-size packet of $VL_k$:

$$L_{Trmax(k)} = \frac{(O_{Eth} + Lmax_k) \cdot 8}{N_{bw}} \quad (10)$$

### 5.1.2. Scheduling of virtual links, $L_{VL}$ (step 1)

For the analysis of a message sent across a virtual link, we assume that the technological latency on transmission specified in the ARINC 664-Part 7 standard is counted just once for all the messages to be transmitted in an uninterrupted sequence. This can be justified because the activity of the end system causing the transmission latency is concurrent with the actual transmission.

So, in this case the latency of a message from stream $\sigma_i$ being sent through $VL_k$ due to the scheduling of the virtual links in a specific processor can be calculated as follows:

$$L_{VL(ik)} = L_{VLQ(ik)} + I_{VL(ik)} \quad (11)$$

where, $I_{VL(ik)}$ is the worst-case interference from the messages of the other VLs in the same processor generating message stream $\sigma_i$ , and $L_{VLQ(ik)}$ is the worst-case latency in the $VL_k$ queue, including the effects of the messages that can be awaiting on $VL_k$.

To obtain the $L_{VLQ(ik)}$ latency we will create a worst-case scenario in which, when the message under analysis is released, the VL buffer already contains the worst-case amount of packets that can interfere the transmission. Therefore we need to calculate the interference of the rest of the messages sharing the VL and also the interference of the previous packets of the message under analysis. For this purpose we take into account the following observations:

- We analyze the messages in a worst-case busy period. A busy period is defined as an interval of time during which the VL queue is not empty. Since the VL is designed to be able to handle its worst-case throughput, the utilization is smaller than 100% and this ensures that there will be time instants at which the VL queue is empty and, therefore, busy periods are bounded. The worst case busy period is created by releasing all the messages from all the message streams of the virtual link at the same time, after having experienced their maximum jitter, and with all subsequent messages with the smallest jitter that makes them arrive within the busy period. This ensures the maximum amount of work concentrated towards the start of the busy period and leads to the worst case.
- Each packet in the queue that is ahead of the message under analysis contributes with an interference equal to the $BAG_k$.
- In addition, each packet of the message under analysis before the last one also contributes with an interference equal to the $BAG_k$.
- A message in the FIFO queue can not be preempted, so when calculating the interference of the rest of the messages in the VL, we only need to consider those that arrived at the queue before the message under analysis (thus excluding the message instance under analysis). If there are $q$ message instances in the queue, we will take into account the interference of the ($q$-1) previous instances.
- Furthermore, the analysis technique should be applied for all the message instances that can be in the queue in the worst case busy period, similarly to the analysis of non-preemptive messages that can be found in [4].

Since the virtual link uses a FIFO queueing discipline, we can analyze the latencies using response time analysis. We model the latency of each message as execution time and we calculate the interference for the first packet of the $q$-th instance of a message from stream $\sigma_i$ to reach the VL scheduler, as follows:

$$w_i(q) = (q-1) \cdot (p_i \cdot BAG_k) + \sum_{j \in MS(VL_k)} \left( \left\lfloor \frac{J_j + (q-1) \cdot T_i}{T_j} \right\rfloor + 1 \right) \cdot (p_j \cdot BAG_k) \quad (12)$$

where, $p_i$ is the worst-case number of packets, $MS(VL_k)$ is the set of message streams that share $VL_k$ with message

stream $\sigma_i$ (excluding itself), $T_j$ and $J_j$ are the period and release jitter of message stream $\sigma_j$. The first term in the equation corresponds to the interference of previous instances of the message stream under analysis, and the second term is the interference by all those messages from other streams that have arrived at the VL queue before the message under analysis. The result of this equation, $w_i(q)$, is the worst-case latency for the *q-th* instance of message stream $\sigma_i$ to reach the VL scheduler after a critical instant.

Eq. (12) is applied for all values of $q$ equal to 1,2,3,…, finishing at $q=Q_i$, where $Q_i$ is the number of instances of message stream $\sigma_i$ that become ready for transmission before the end of the busy period. The number of instances is calculated as indicated in [4]:

$$Q_i = \left\lceil \frac{J_i + BP_k}{T_i} \right\rceil \quad (13)$$

where $BP_k$ is the length of the busy period for any message of $VL_k$ (note that since the queue is FIFO it does not depend on the particular message stream), and it is given by the following recurrence relation, starting with an initial value of $BP_k^0 = BAG_k$, and finishing when $BP_k^{n+1} = BP_k^n$:

$$BP_k^{n+1} = \sum_{j \in MU(VL_k)} \left\lceil \frac{J_j + BP_k^n}{T_j} \right\rceil \cdot (p_j \cdot BAG_k) \quad (14)$$

where, $MU(VL_k)$ is the set of message streams using $VL_k$ (including message stream $\sigma_i$).

Using the results obtained for the different values of $q$ in (12), the worst-case latency for the last packet of the $q$-th instance of message stream $\sigma_i$ due to the messages that can be waiting on the VL queue can be calculated in the following way:

$$L_{VLQ(ik)} = \max_{q = 1, 2, \ldots, Q_i} [L_{VLQ(ik)}(q)] \quad (15)$$

where,

$$L_{VLQ(ik)}(q) = w_i(q) + (p_i - 1) \cdot BAG_k - (q-1) \cdot T_i \quad (16)$$

In Eq. (16), we include a delay corresponding to the maximum number of packets minus one, which is the time that the last packet of the *q-th* message instance must wait for transmission.

The interference of the rest of VLs, $I_{VL(ik)}$, can be calculated based on the formula indicated in [1] as follows:

$$I_{VL(ik)} = L_T + \frac{\sum_{j \in S_k} ((O_{Eth} + Lmax_j) \cdot 8)}{N_{bw}} \quad (17)$$

with

$$I_{VL(ik)} + \frac{(O_{Eth} + Lmax_k) \cdot 8}{N_{bw}} - L_{Tmin} \leq 500\,\mu s \quad (18)$$
$$(L_T = L_{Tmin} + J_{Tech}) \leq 150\,\mu s$$

where, $S_k$ is the set of VLs in the same processor than $VL_k$ (excluding it). The requirements for $I_{VL(ik)}$ specified in the ARINC 664 Part 7 specification (see subclause 1.2.4.3 “Jitter” in attachment 1 to [1]) and expressed in Eq. (18) should be taken into account when we are parameterizing the application, in particular in the assignment of the number of VLs and their *Lmax* parameters.

Figure 3 shows an example for illustrating the calculation of latencies in the VL scheduler of an end system used in transmission. We assume that there are two virtual links, $VL_1$, and $VL_2$. Two periodic message streams, $M_1$ and $M_2$ share $VL_1$, while a third message stream, $M_3$, uses $VL_2$. The following tables show the configuration of the VLs and the message sizes, the number of packets and packet transmission times, assuming a 100Mb/s wire and the packet overhead of 67 bytes

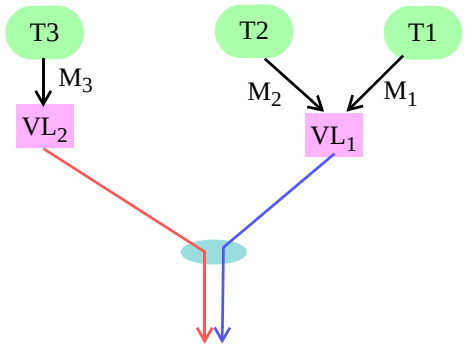


Figure 3. Example with 2 VLs and 3 message streams

**TABLE 1.**Configuration of VLs

| VL | BAG (μs) | Lmax (bytes) |
|---|---|---|
| VL1 | 16000 | 200 |
| VL2 | 16000 | 1000 |

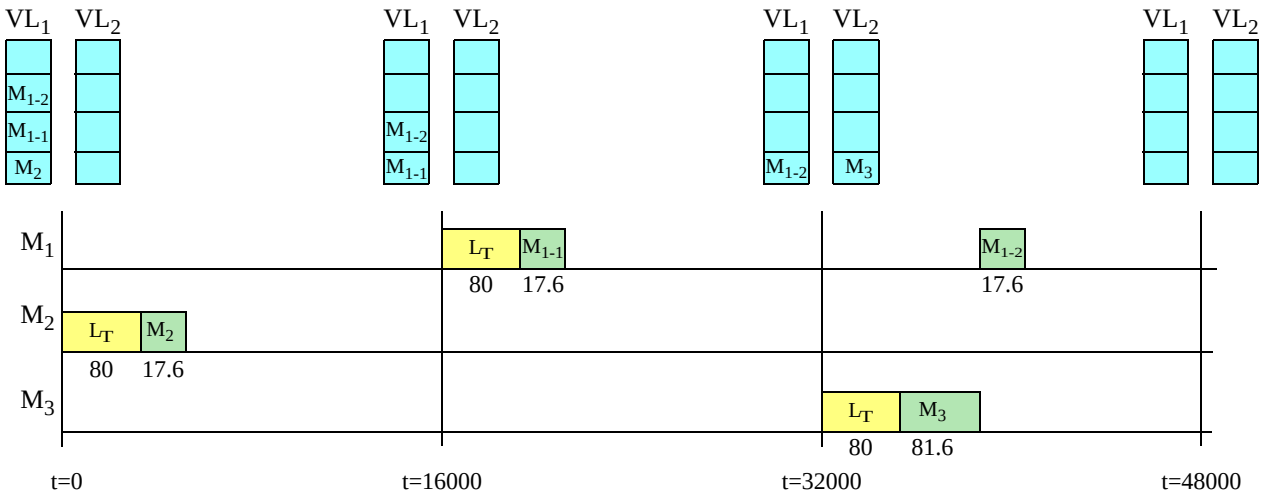


Figure 4. Time Diagram showing the latencies in the VL scheduler, in μs (for visibility, they are not scaled)

**TABLE 2.** Message requirements (times in μs)

| Message | VL | Ji | Li | Ti | Num pckts | $L_{Tr(i)}$ |
|---|---|---|---|---|---|---|
| M1 | VL1 | 0 | 306 | 50000 | 2 | 17.6 |
| M2 | VL1 | 0 | 153 | 100000 | 1 | 17.6 |
| M3 | VL2 | 0 | 953 | 200000 | 1 | 81.6 |

Figure 4 shows a time diagram of a worst-case scenario that starts with the $VL_1$ queue having received an instance of $M_2$ and then an instance of $M_1$, fragmented into two packets, $M_{1\text{-}1}$, and $M_{1\text{-}2}$. We assume LT=80 μs. The figure also shows the state of the VL queues at the start of each BAG period. $M_{1\text{-}2}$ has to wait for two BAGs required to send $M_2$ and $M_{1\text{-}2}$ (total time= 2x16ms=32ms corresponding to the $L_{VLQ(ik)}$ term). In addition, before it is sent it suffers the technological latency in the end system[1] ($L_T$=80μs) and the impact from messages of other VLs, in this case $M_3$ (81.6μs). Both terms contribute to $I_{VL(ik)}$. Therefore the total latency for the full $M_1$ message is in the worst case 32000+80+81.6+17.6 = 32179.2 μs.

From the analysis, we can derive the following observations that may be useful to the designer:

- For the proposed analysis the following two situations are equivalent: different tasks using different ports all of them attached to one VL, or different tasks using one port that is attached to one VL. The reason is that we are not able to distinguish the originating port when the message comes into the VL queue fragmented in packets. We assume that packets of the same message are enqueued in consecutive positions of the queue, i.e., packets from other messages can not be enqueued in the middle.
- The exclusive use of a VL by an application task can lead to a high number of VLs, which may make it difficult to meet the latency requirements.
- Sharing a VL by several applications tasks makes it easier to meet the latency requirements, but a message can suffer a high interference due to the rest of messages sharing the VL ($I_{VLQ}$).

### 5.1.3. Switch management, $L_{Sw}$ (step 3)

The latency due to the switch management is composed of two terms:

- the latency to deliver a packet from the incoming to the outgoing port, which can be considered as the hardware latency provided by the manufacturer,
- and the interference due to the rest of the packets sent to the same destination end system.

We can calculate the total latency in the switch for the last packet of message stream $\sigma_i$ being sent through $VL_k$ as (assuming that the utilization of the output link is under 100%):

$$L_{Sw(ik)} = L_S + L_{SQ(ik)} \quad \textbf{(19)}$$

where, $L_S$ is the switch hardware latency and $L_{SQ(ik)}$ is the interference of the rest of the packets in the output queue associated with the outgoing port of $\sigma_i$ in the switch.

To obtain the $L_{SQ(ik)}$ latency we can apply a similar approach as we used for calculating $L_{VLQ(ik)}$. We will create a worst-case scenario in which, when the packet

1. Recall that we assume that this $L_T$ latency is only charged once per BAG, as this is the implicit assumption in the equations that appear in subclause 1.2.4.2 of Attachment 1 to [1].

under analysis is enqueued into the output queue, it already contains the worst-case amount of packets that can interfere the transmission. Therefore we need to calculate the interference of the rest of the packets sent through this outgoing port and also the interference of the previous packets coming from the same VL of the message stream $\sigma_i$ under analysis. For this purpose we take into account the following observations:

- We analyze the packets in a worst-case busy period. In this case, a busy period is defined as an interval of time during which the output queue is not empty. The worst-case busy period is obtained after a critical instant created with the same criteria as in the analysis of $L_{VLQ(ik)}$.
- Each packet in the output queue that is ahead of the packet under analysis contributes with an interference equal to its worst-case transmission time on the physical link, that can be calculated using Eq. (10) with the maximum packet length for the corresponding virtual link.
- In addition, each packet of the message under analysis before the last one also contributes with an interference equal to the transmission time for a worst-case size packet, calculated using Eq. (10) for $VL_k$.
- A packet in the FIFO queue can not be preempted, so when calculating the interference of the rest of the packets in the output queue, we only need to consider those that arrived at the queue before the packet under analysis (thus excluding itself).
- Furthermore, the analysis technique should be applied for all the packets that can be in the queue in the worst case busy period, similarly to the analysis of non-preemptive messages that can be found in [4].

We model the latency of each packet as execution time and we calculate the interference for the $q$-th packet coming from $VL_k$ to reach the physical output link, as follows:

$$w_k(q) = (q-1)\cdot L_{Trmax(k)} + \sum_{j \in DP(VL_k)} \left( \left\lfloor \frac{Jp_j + (q-1)\cdot BAG_k}{BAG_j} \right\rfloor + 1 \right) \cdot L_{Trmax(j)} \quad (20)$$

where $DP(VL_k)$ is the set of VLs that have as destination port the outgoing port of $VL_k$ , excluding itself; and $Jp_j$ is the worst-case release jitter of the packets coming from $VL_j$, and is calculated by adding the output jitter of the packets at the source end systems, and the jitter to deliver a packet from the incoming to the outgoing port, as follows:

$$Jp_j = J_{Tech} + \sum_{m \in S_j} L_{Trmax(m)} + (L_S - L_S^b) \quad (21)$$

where, $S_j$ is the set of VLs in the same processor than $VL_j$ (excluding it).

The first term in Eq. (20) corresponds to the interference of previous packets of $VL_k$, and the second term is the interference by all those packets from other VLs. The result of this equation, $w_k(q)$, is the worst-case latency for the $q$-*th* packet of $VL_k$ to reach the output physical link after a critical instant.

Eq. (20) is applied for all values of $q$ equal to 1,2,3,…, finishing at $q=Q_k$, where $Q_k$ is the number of packets of $VL_k$ that become ready for transmission before the end of the busy period. The number of packets is calculated as indicated in [4]:

$$Q_k = \left\lceil \frac{Jp_k + BP_k}{BAG_k} \right\rceil \quad (22)$$

where $BP_k$ is the length of the busy period in the output port of $VL_k$, and it is given by the following recurrence relation, starting with an initial value of $BP_k^0 = L_{Trmax(k)}$, and finishing when $BP_k^{n+1} = BP_k^n$:.

$$BP_k^{n+1} = \sum_{j \in DP(VL_k) \cup k} \left\lceil \frac{Jp_j + BP_k^n}{BAG_j} \right\rceil \cdot L_{Trmax(j)} \quad (23)$$

Using the results obtained for the different values of $q$ in (20), the worst-case latency for the last packet of the $q$-th instance of message stream $\sigma_i$ due to the packets that can be waiting on its associated output queue can be calculated in the following way:

$$L_{SQ(ik)} = \max_{q = 1, 2, \ldots, Q_k} [L_{SQ(ik)}(q)] \quad (24)$$

where,

$$L_{SQ(ik)}(q) = w_k(q) - (q-1)\cdot BAG_k \quad (25)$$

### 5.1.4. Message management at destination end system, $L_{Rec}$ (step 5)

We can assume that this latency is equal to the technological latency in the reception $L_{Rec} = L_R$ .

### 5.1.5. Best-case latencies

In order to calculate the output jitter of the messages sent through the network it is necessary to calculate best-case latencies in addition to the worst case values.

*Steps 2 and 4: Transmission of the last packet to the switch or to the end system*

The best-case number of packets of a message belonging to stream $\sigma_i$ being sent through $VL_k$ can be calculated as:

$$p_i^b = \left\lceil \frac{M_i^b}{Lmax_k - O_{Prot}} \right\rceil \quad (26)$$

where $O_{prot}$ is the protocol overhead in bytes.

The size of the last packet of a message belonging to stream $\sigma_i$ being sent through $VL_k$ can be obtained for the best case by applying Eq. (6) to the best payloads for this packet, $Np^b_{i,last}$. This payload can be calculated as follows:

$$Np^b_{i,\,last} = M_i^b - (p_i^b - 1) \cdot (Lmax_k - O_{Prot}) \quad (27)$$

This equation calculates the number of packets by subtracting an integer number of maximum-size payloads from the message payload. It could be argued that for calculating the best case a minimum payload of size one can be generated if all the previous packets fill in their maximum payload. However, this would not lead to a best-case latency, since we would be producing one more packet than is necessary, and the latency of a full packet, equal to the BAG, is much larger than the transmission latency.

Using the best-case payload of the last packet we can calculate its total size using Eq. (6):

$$N^b_{i,\,last} = O_{Eth} + N_{min} \qquad Np^b_{i,\,last} \in [1,17] \quad (28)$$
$$N^b_{i,\,last} = O_{Eth} + O_{Prot} + Np^b_{i,\,last} \qquad Np^b_{i,\,last} \in [18,1471]$$

And then we can calculate the best-case latency of a last packet transmitted through the Ethernet link applying Eq. (5) with this size:

$$L^b_{Tr(i)} = \frac{N^b_{i,\,last} \cdot 8}{N_{bw}} \quad (29)$$

*Step 1: Scheduling of virtual links*

The latency of a message from stream $\sigma_i$ sent through $VL_k$ due to the scheduling in the virtual link can be calculated as the sum of $L^b_{VLQ(ik)}$, which is the best-case latency due to the messages that can be awaiting on $VL_k$; and $I^b_{VL(ik)}$, and is the best-case interference from the messages of the other VLs in the same processor generating message stream $\sigma_i$:

$$L^b_{VL(ik)} = L^b_{VLQ(ik)} + I^b_{VL(ik)} = (p_i^b - 1) \cdot BAG_k + L_{Tmin} \quad (30)$$

A lower bound on the $L^b_{VLQ(ik)}$ latency can be calculated assuming that there are no messages to be sent in the VL except for the message under analysis, which has its minimum payload. In the best case this is a number of *BAG*s equal to the minimum number of packets minus one. For the $I^b_{VL(ik)}$ latency, we use the minimum technological latency defined in the ARINC-644, Part 7 standard.

*Step 3: Switch management*

A lower bound on the best case management latency can be obtained by assuming that there is no contention from other messages inside the switch, and therefore we just take into account the minimum hardware latency of the switch latency that we call $L^b_S$.

*Step 5: Message management at destination end system*

We can assume that this latency is equal to the best technological latency in the reception, which is $L^b_{Rec} = L^b_R$.

### 5.1.6. Total latency

The worst-case latency, $L_{ik}$, for a message stream $\sigma_i$ sent through virtual link $VL_k$ can be calculated as the sum of latencies of steps 1 through 5 plus its own input jitter:

$$L_{ik} = L_{VL(ik)} + 2 \cdot L_{Tr(i)} + L_{Sw(ik)} + L_{Rec} + J_i \quad (31)$$

Similarly, we calculate the best-case latency, in the following way:

$$L^b_{ik} = [(p_i^b - 1) \cdot BAG_k + L_{Tmin}] + 2 \cdot L^b_{Tr(i)} + L^b_S + L^b_{Rec} \quad (32)$$

The output jitter is the difference between the worst-case and the best-case latencies.

## 5.2. Analysis for two or more switches and for multicast messages

When a message has to cross two or more switches to reach the destination end system, the latency due to the management of each switch and one extra transmission for each switch should be added. Figure 5 shows the communication process with multiple switches. In this case, the worst-case latency for the message stream $\sigma_i$

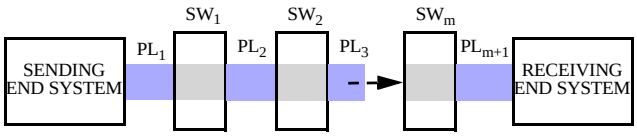


Figure 5. Communication process with *m* switches

being sent through $VL_k$ and crossing *m* switches can be calculated as follows:

$$L_{ik} = L_{VL(ik)} + \sum_{\forall link(i)} L_{Tr(i)} + \sum_{\forall switch(i)} L_{Sw(ik)} + L_{Rec} + J_i \quad \textbf{(33)}$$

where *link*(*i*) is the set of *m*+1 physical links traversed by message stream $\sigma_i$ and *switch*(*i*) is the set of *m* switches traversed by $\sigma_i$. We assume that different link speeds and switches may be used.

A similar approach can be followed to obtain the best-case latency for the message stream $\sigma_i$ being sent through $VL_k$ :

$$L_{ik}^{b} = [(p_i^b - 1) \cdot BAG_k + L_{Tmin}] + \sum_{\forall link(i)} L_{Tr(i)}^{b} + \sum_{\forall switch(i)} L_S^b + L_{Rec}^b \quad \textbf{(34)}$$

The analysis presented in this section has focused, for simplicity of presentation, on messages with just one single destination. However, the analysis works without modification for multicast messages. For these messages, the latency of each destination has to be calculated. The latencies for Steps 1 and 2 in the communication process are calculated in the same way as for unicast messages. Step 3 has to be repeated for every output port queue in the switch. Step 4 has to be repeated using the characteristics of the corresponding output link, and Step 5 is also repeated in each of the destination end systems. If one or more of the paths of the message traverse several switches, then the analysis for multiple switches is done for each of these paths.

## 6. Combined analysis of the distributed system

Offset based response-time analysis techniques exist [6][9] and they can be combined with the developed response time analysis for AFDX networks. To analyze a distributed application it is necessary to integrate both analyses.

One of the interesting properties of offset-based response time analysis is that it provides a natural way of composing analysis in different resources using different scheduling policies. The analysis in each resource is made independently, and therefore we can use whatever technique is appropriate. As a result of the analysis in one resource we get response times and jitter terms than can be used to calculate equivalent offsets and jitters for the analysis in the other resources. In this way we can combine techniques for fixed priorities, dynamic priorities, (EDF), time partitioned scheduling, and AFDX communication.

To make this integration effective we just need to explain how to calculate response times and jitters from the latencies obtained in the AFDX network, and how to calculate the release jitters for the messages in the network. Suppose the message stream $\sigma_i$ shown in Figure 6, sent at the at the finalization of task $\tau_{aj-1}$ and activating, in turn, $\tau_{aj+1}$ in its end-to-end flow $\Gamma_a$. Task $\tau_{aj}$ is just the model of the $\sigma_i$ message in the end-to-end flow.

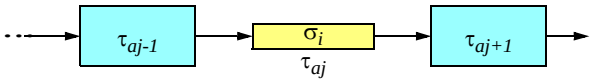


Figure 6. Portion of an end-to-end flow with a message stream sent through an AFDX network

The worst-case release jitter of the message stream, $J_i$, is obtained as the difference between the worst and the best case response time of $\tau_{aj-1}$ :

$$J_i = R_{ij-1} - R_{ij-1}^{b} \quad \textbf{(35)}$$

The worst and best-case latencies of the AFDX message shown in equations (33) and (34) already take into account this jitter, and are relative to the best possible release time , which is in this case $R_{ij-1}^{b}$. Therefore, the worst and best-case response times of $\sigma_i$ (or $\tau_{aj}$) are obtained as:

$$\Phi_{ij,max} = R_{ij} = R_{ij-1}^{b} + L_{ik}$$
$$\Phi_{ij,min} = R_{ij}^{b} = R_{ij-1}^{b} + L_{ik}^{b} \quad \textbf{(36)}$$

where *k* is the index of the VL through which message stream $\sigma_i$ is sent. From these values we would calculate inherited offsets and jitters that could be used in the Holistic or Offset-Based response time analysis algorithms.

To complete the integration of the analysis in distributed systems we must take into account the clock synchronization mechanisms, and their precision, if present. In a distributed end-to-end flow that is activated from a synchronized workload event, the tasks that are assigned to synchronized processors are all synchronized tasks. All the other tasks that execute in non-synchronized processors are considered non synchronized.

We have to include the precision of the clock synchronization mechanism in the analysis of synchronized tasks. In a given end-to-end flow, every time we cross from one processor to another we need to add an additional jitter term equal to the precision of the clock synchronization. This is added to the initial $J_{ij}$ release jitter term of the destination task in the flow.

For example, if the end-to-end flow in Figure 6 is synchronized and the processor clocks are also synchronized, we need to use an initial jitter for $\tau_{aj+1}$ of $J_{aj+1}$+*precision*(*clock*). This approach will take into consideration the errors in the synchronization of the clocks.

## 7. Case study

This section shows a simple case study that is used to illustrate the analysis in an AFDX switch. It contains two situations: the first situation has 4 message streams, two of them sharing the same virtual link; the second one has also 4 message streams, and are sent through different virtual links.

This case-study contains an application with 8 tasks allocated in 7 partitions and 3 processors. Four of these tasks produce non-synchronized messages. Table 3 shows the relevant characteristic of this set of tasks (times in milliseconds). Initial input jitter for the end-to-end flows is assumed to be zero. All tasks and non-synchronized.

**TABLE 3.**Task set for the case-study

| Task | Proc. | Part. | Ci | Ti |
|---|---|---|---|---|
| T1 | CPU1 | P1 | 10 | 50 |
| T2 | CPU1 | P1 | 10 | 100 |
| T3 | CPU1 | P2 | 2 | 20 |
| T4 | CPU2 | P3 | 10 | 40 |
| T5 | CPU3 | P4 | - | - |
| T6 | CPU3 | P5 | - | - |
| T7 | CPU2 | P6 | - | - |
| T8 | CPU3 | P7 | - | - |

We are assuming that the value of $L_T^b$ is 80 μs, $L_{Tmin}^b$=$J_{Tech}$= 40 μs., $L_S$ = 100μs, $L_S^b$ = 70μs , $L_{Rec}^b$=40 μs.

### 7.1. Situation 1

In this situation tasks T1, T2, T3 and T4 send messages at the end of their executions with the parameters shown in Table 4 (times in milliseconds and lengths of messages in bytes). Messages from T1 and T2 share Virtual Link VL1. Task T3 and T4 transmit through Virtual Links VL2 and VL3 respectively. The destination end system for VL1 and

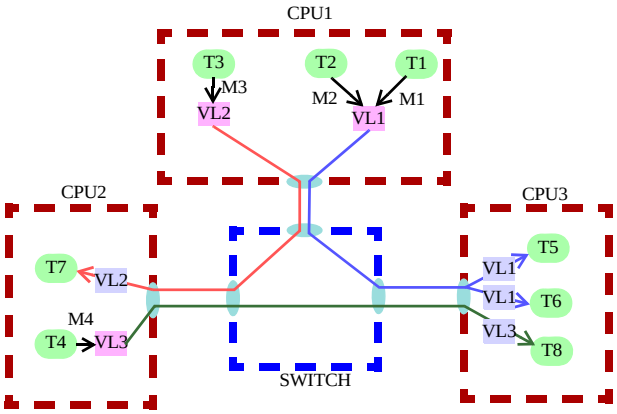


Figure 7. Diagram for Situation 1

VL3 is processor CPU3. The destination end system for VL2 is processor CPU2. The release jitter of each message is produced by the variability of the execution of the task that generates it. The lengths of the messages are fixed (the best and the worst sizes are equal), and we have chosen to have packets of size equal to the *Lmax* value of their respective VLs.

**TABLE 4.**Message set for Situation 1

| Msg | VL | Task (Send) | Task (Rec.) | Ji | Li | Ti |
|---|---|---|---|---|---|---|
| M1 | VL1 | T1 | T5 | 20 | 306 | 50 |
| M2 | VL1 | T2 | T6 | 60 | 153 | 100 |
| M3 | VL2 | T3 | T7 | 5 | 953 | 20 |
| M4 | VL3 | T4 | T8 | 15 | 453 | 60 |

Virtual Links have been configured to support the bandwidth necessary to send the messages under their control. Table 5 shows the *BAG* and *Lmax* parameters for each VL, as well as the connections in the switch (only processors are indicated for simplicity). Figure 7 shows the diagram of the system for Situation 1 with its tasks, messages, VLs and switch connections.

**TABLE 5.**Configuration of VLs for Situation 1

| VL | BAG | Lmax | SW-in | SW-out |
|---|---|---|---|---|
| VL1 | 16 | 200 | CPU1 | CPU3 |
| VL2 | 16 | 1000 | CPU1 | CPU2 |
| VL3 | 32 | 500 | CPU2 | CPU3 |

The difference between the worst and the best cases for the latencies of M1 results in the contribution to the output jitter generated to the task T5 receiving this message (this jitter is 16.2132 milliseconds).

We have developed a tool to calculate automatically the latency in the AFDX network. The results obtained for the analysis of this example are shown in Table 6 (times in milliseconds).

**TABLE 6.**Results of the analysis for Situation 1

| Message | VL | Li | Lib | Input Ji | Output Ji |
|---|---|---|---|---|---|
| M1 | VL1 | 32.3984 | 16.1852 | 20 | 36.2132 |
| M2 | VL1 | 32.3984 | 0.1852 | 60 | 92.2132 |
| M3 | VL2 | 0.4208 | 0.3132 | 5 | 5.1076 |
| M4 | VL3 | 0.3480 | 0.2332 | 15 | 15.1076 |

## 7.2. Situation 2

In this situation (see Figure 8 and Table 7) messages from stream M2 are sent through the new Virtual Link VL4, which is different from the one used by M1.

**TABLE 7.**Message set for Situation 2

| Message | VL | Task (Send) | Task (Rec.) | Ji | Li | Ti |
|---|---|---|---|---|---|---|
| M1 | VL1 | T1 | T5 | 20 | 306 | 50 |
| M2 | VL4 | T2 | T6 | 60 | 153 | 100 |
| M3 | VL2 | T3 | T7 | 5 | 953 | 20 |
| M4 | VL3 | T4 | T8 | 15 | 453 | 60 |

The configuration of the Virtual Links is shown in Table 8. The configuration of the Virtual Link VL1 has changed to accommodate only the traffic for M1, while the traffic of M2 goes to VL4.

In this case the contribution of message M1 to the output jitter, i.e., the difference between the worts and best case latencies, is 0.2484 milliseconds, which is shorter than for Situation 1, and with a shorter worst-case latency. This is mainly because M1 is sent in one packet.

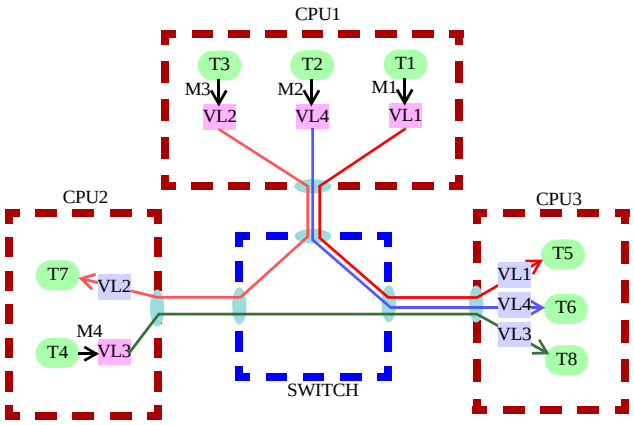


Figure 8. Diagram for Situation 2

**TABLE 8.**Configuration of VLs for Situation 2

| VL | BAG | Lmax | SW-in | SW-out |
|---|---|---|---|---|
| VL1 | 32 | 353 | CPU1 | CPU3 |
| VL2 | 16 | 1000 | CPU1 | CPU2 |
| VL3 | 32 | 500 | CPU2 | CPU3 |
| VL4 | 64 | 200 | CPU2 | CPU3 |

The results obtained when applying the analysis with the developed tool for this example are shown in Table 9 (times in milliseconds)..

**TABLE 9.**Results of the analysis for Situation 2

| Msg | VL | Li | Lib | Input Ji | Output Ji |
|---|---|---|---|---|---|
| M1 | VL1 | 0.45808 | 0.20968 | 20 | 20.2484 |
| M2 | VL4 | 0.45808 | 0.1852 | 60 | 60.27288 |
| M3 | VL2 | 0.45064 | 0.3132 | 5 | 5.13744 |
| M4 | VL3 | 0.37064 | 0.2332 | 15 | 15.13744 |

We can see how M2 has now a shorter worst-case latency than for Situation 1 due to the fact that for Situation 2, M1 and M2 do not share a virtual link.

# 8. Conclusions and future work

In this paper we have developed a new analysis response time analysis technique for AFDX networks. This analysis technique can be combined with other response time analysis techniques to analyze distributed systems.

Prototype tools have been developed to assist us in checking the analysis techniques. They have been used to analyze the presented case study. As future work, the new technique will be added as an extension to the open-source MAST model and toolset for real-time applications [7].

As future work we plan to adapt the analysis to the current version of the standard. The 2003 version of the standard was used in this work and the 2005 or further version should be used. In particular, it is important to study the effects of the token bucket algorithm that appears in the 2005 version of the ARINC 664-Part 7 standard. This algorithm is used in the AFDX Switch to reject “non-conformant” incoming traffic, and also for “Traffic Shaping” the outgoing messages in the End-System. Also important is to take into consideration the presence of two priority levels inside an AFDX Switch.

In addition, we plan to do an evaluation and validation of the new analysis by comparison with actual latencies in real or simulated AFDX hardware.

Other planned extensions are the support for subVLs, and the extension of offset-based analysis techniques to

analyze synchronized message streams in the AFDX network.

## References


[1] Airlines Electronic Engineering Committee, Aeronautical Radio INC., “ARINC Specification 664: Aircraft Data Network, Part 7 - Deterministic Networks”, October 2003.

[2] ARINC. “Avionics Application Software Standard Interface”. ARINC Specification 653-1. March 2006.

[3] Condor Engineering, “AFDX/ARINC 664 tutorial,” May 2005.
http://www.acalmicrosystems.co.uk/whitepapers/sbs8.pdf

[4] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien, “Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised,” Journal of Real-Time Systems, 35 (3), Springer, pp. 239-272, 2007.

[5] M. González Harbour, J.J. Gutiérrez García, J.C. Palencia Gutiérrez, and J.M. Drake Moyano. “MAST: Modeling and Analysis Suite for Real Time Applications”. Proceedings of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, IEEE Computer Society Press, pp. 125-134, June 2001.

[6] Jukka Maki-Turja and Mikael Nolin. “Efficient implementation of tight response-times for tasks with offsets”. Real-Time Systems Journal, 40(1):77–116, February 2008.

[7] MAST: Modelling and Analysis Suite for Real-Time Systems. Home page:
http://mast.unican.es

[8] J.C. Palencia, and M. González Harbour, “Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems”. Proceedings of the 20th IEEE Real-Time Systems Symposium, 1999.

[9] J.C. Palencia and M. González Harbour, “Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF”. Euromicro conference on real-time systems, Porto, Portugal, June 2003.

[10] Ruggedcom Industrial Strength Networks, “Latency on a Switched Ethernet Network,” April 2008.
http://www.ruggedcom.com/pdfs/application_notes/latency_on_a_switched_ethernet_network.pdf

[11] M. Spuri. “Holistic Analysis of Deadline Scheduled Real-Time Distributed Systems”. RR-2873, INRIA, France, 1996.

[12] K. Tindell, and J. Clark, “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”. Microprocessing & Microprogramming, Vol. 50, Nos.2-3, pp. 117-134, April 1994.