Response time analysis in AFDX networks with sub-virtual links and prioritized switches

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 policing 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 the contention in the end-systems and in the switches. This analysis 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 heterogeneous distributed systems.

1. Introduction

AFDX (Avionics Full Duplex Switched Ethernet) is a communications network defined in the ARINC-644, Part 7 standard [2] 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. The AFDX switches are capable of filtering non conformant traffic, and can do traffic policing based on two priority levels. In addition, AFDX provides two redundant hardware communication links for fault-tolerant operation.

AFDX [9][20] defines the communication process among end-systems (processing nodes) where bandwidth and bounded latency are guaranteed. A flow of communication packets between two end systems suffers bounded jitter that depends on the global network traffic at a given time.

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 integrated in the MAST modelling and analysis suite [11][15]. From that model, we have developed a response time analysis for AFDX networks that can be integrated with the response-time analyses in other resources, using the heterogeneous RTA techniques described in [19]. In this way we can perform a holistic end-to-end response time analysis in 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 model of the AFDX network and our assumptions for the analysis. Related work is presented in Section 4. 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, and also shows the highlights of the proposed holistic technique. Finally, Section 8 presents the conclusions and future work.

2. The AFDX Network

The usual way for applications to exchange messages in AFDX is through the communication ports defined in the ARINC 653 standard [3], 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. 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:

1. This work has been funded in part by the Spanish Government under grants TIN2008-06766-C03-03 (RT-MODEL) and TIN2011-28567-C03-02 (HI-PARTES), and by the European Commission under project FP7/NoE/214373 (ArtistDesign).

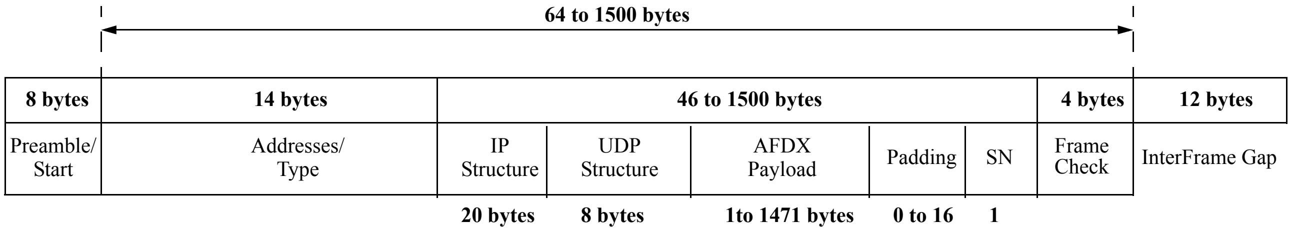

Figure 1. Structure of the Ethernet frame with an AFDX payload

- *Sampling Port*: the arriving message overwrites the current message stored in the buffer.
- *Queueing Port*: the arriving message is appended to a FIFO queue.

Queueing ports are required to manage at least 8Kbytes of data and allow messages to be fragmented into several packets, while sampling ports are limited to single-packet messages.

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* (VL) defined (see 3.2.1 "Virtual Link" in [2]) 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_i is characterized by two parameters used for traffic regulation:

- $Lmax_i$: the largest Ethernet frame, which is a value in bytes.
- BAG_i: the Bandwidth Allocation Gap, 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. In a partitioned system using ARINC 653, several AFDX communication ports may share the same VL to transmit their packets as long as they belong to the same partition. Sharing 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.

Since message fragmentation is not allowed for sampling ports, we need 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. When very long packets could excessively delay the transmission of other contending messages with very short deadlines it is possible to adjust the *Lmax* value forcing fragmentation of the long messages.

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 with the packets 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 3.2.4.3 "Jitter" in [2]).

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

$$MaxJitter \le 40\mu s + \frac{\sum_{i \in \{set\ of\ VLs\}} ((20 + Lmax_i) \cdot 8)}{N_{bw}}$$
$$MaxJitter \le 500\mu s \tag{1}$$

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 allowed jitter is 500 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 (*sub-VL*s), all of them having a dedicated FIFO queue which is read on a round-robin basis by the VL scheduler. The round robin algorithm works over IP fragmented packets.

The ARINC 664 specification [2] 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 3.2.4.3 "Jitter" in [2]):

- 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:

$$MaxLatency \le BAG + MaxJitter + L_T \tag{2}$$

  where, L_T is the technological latency in the transmission, defined as the time required for a packet to traverse 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 3.2.4.1 "Latency" in [2]) irrespective of whether one or more messages are sent.

- For those messages requiring fragmentation into *p* packets, 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:

$$MaxLatency(p) \le p \cdot BAG + MaxJitter + L_T \tag{3}$$

Once a packet is ready to be transmitted, it is sent to the switch using the full capacity of the physical link. If the switch detects that the input traffic does obey to the bandwidth restrictions of the VLs, it will filter spurious or non-conformant messages to guarantee that the rest of the traffic remains schedulable. In this paper we assume that the bandwidth restrictions are obeyed. As a consequence, no message filtering is needed and non-conformant traffic is not considered for the response time analysis, which assumes a correct operation of all the end systems.

The switch delivers correct packets 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. The latency introduced when a packet is delivered from the incoming to the outgoing port, known as the hardware latency of the switch, L_S, must also be taken into account in the analysis. It should be less than 100μs.

A new contention point, and a new source of jitter appears in the queue where packets wait to be sent to the destination end system. According to the AFDX specification, the VLs can be configured with two priorities. The output port queues are priority queues where messages are enqueued with either high or low priority. The priority level is defined in a configuration table on a VL basis. Messages of the same priority are kept in FIFO order. We will study the contention effects of these queues in the analysis.

Once the packet is ready to be transmitted from the outgoing port queue, 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 if it is not read at a sufficient rate. Similar to the L_T value, the technological latency of the end system in reception, L_R, should be bounded and lower than 150μs.

3. System model and assumptions

The objective of the analysis 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 ready to be 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.

We will assume that 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 [20], the latency for bits transmitted through a fiber optic link of 100 meters length is around 0.5 μs. For comparison, 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.

We assume that the queues in the switches and in the end systems are large enough to accommodate the worst-case traffic. Our end-to-end response time analysis can be used to ensure that the application removes messages from the queues on time by reading them at the correct rates.

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 jobs can send one message to an AFDX port. The stream of periodic events causes a stream of task jobs to execute, which results in a stream of messages being sent. These messages can, in turn, trigger other tasks. At the end of this sequence of messages and tasks we can impose end-to-end deadlines, which may be larger than the event periods.

We will assume that messages from different tasks are non synchronized, i.e., there is no restriction on the temporal relation between message releases from different tasks.

A message stream σ_i in the AFDX network is characterized by the following parameters:

- M_i, worst-case number of bytes of a message: it is the maximum number of bytes of the message payload. M_i^b, is the best-case number of bytes.
- p_i, the number of packets into which a message is fragmented, for the worst-case analysis. The best-case number of packets is p_i^b.
- Np_i, worst-case number of bytes of a packet payload : it is the maximum number of bytes of the payload of a single packet. Np_i^b is the best-case number of bytes of a packet payload.
- N_i, total worst-case number of bytes of a packet: it is the maximum number of bytes of a single packet, including the message payload (Np_i) and the overhead bytes (both ethernet and protocol overheads). N_i^b is the best-case number of bytes of a packet.
- T_i, period: it is the minimum time between the nominal release of two messages of the σ_i stream. Each message in a stream is nominally released at ϕ_i+nT_i, $n=\{0, 1, 2, 3, ...\}$, where ϕ_i is an arbitrary phase, considered unknown due to the non synchronized nature of the message streams. The actual release of the messages may be affected by jitter (see below).
- J_i, release jitter: it is the time expressing the variability in the release of the messages with respect to the nominal periodic release. It usually depends on the output jitter of the task sending the message. We assume that J_i may be larger than T_i. The actual release time of the messages on stream σ_i is in the interval $[\phi_i + nT_i, \phi_i + nT_i + J_i]$, $n=\{0, 1, 2, 3, ...\}$.
- L_i, worst-case latency, and L_i^b, best-case latency: 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.

There are parameters linked to the hardware which are needed to determine the latency of the complete transmission of a message:

- N_{bw}, speed of the Ethernet link: it is the number of bits per second transmitted through the physical link.
- L_T, maximal technological latency in the AFDX hardware at the sending end system. We obtain it as the sum of two other parameters defined in the AFDX standard:
  - L_{Tmin}, minimum technological latency in the AFDX hardware: this is the minimum value of the sender technological latency.
  - J_{Tech}, the minimum fixed technological jitter: this parameter is defined in the ARINC-644, Part 7 standard (first equation and "note" in subclause 3.2.4.3 "Jitter" in [2]), with a typical value of 40μs. It is the variable part of the sender technological latency.
- L_R, maximal technological latency in reception, defined in the AFDX standard. The best-case value is called L_R^b.
- L_S, maximal switch hardware latency, defined in the AFDX standard. The best-case value is called L_S^b.

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

- O_{Eth}, the Ethernet layer 1 overhead: 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.
- O_{Prot}, protocol overhead: it is the number of overhead bytes corresponding to ethernet layer 2 and upper protocol layers used for communications. Its value is 47 bytes for the UDP/IP used in AFDX.
- F_{min}, minimum Ethernet frame: it is the number of bytes of the minimum Ethernet frame, excluding the ethernet layer 1 overhead, O_{Eth}. 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 will evaluate it

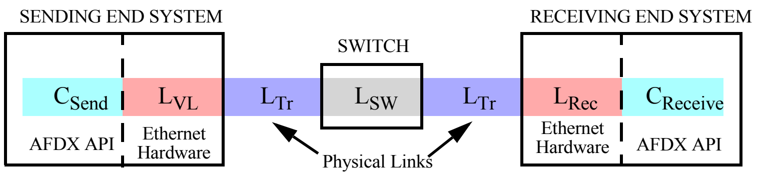

Figure 2. Latencies of the communication process

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 (see Figure 2):

- *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 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 3.2.4.3), 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 last packet of 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. Furthermore, Figure 3 summarizes our view of a distributed application using an AFDX network, which is composed by a set of end systems executing several tasks communicating through one or more switches. The communication process has two kinds of contention points: the VL schedulers at the sending end systems, and the prioritized FIFO queues (high and low priorities) at the output ports of the switches.

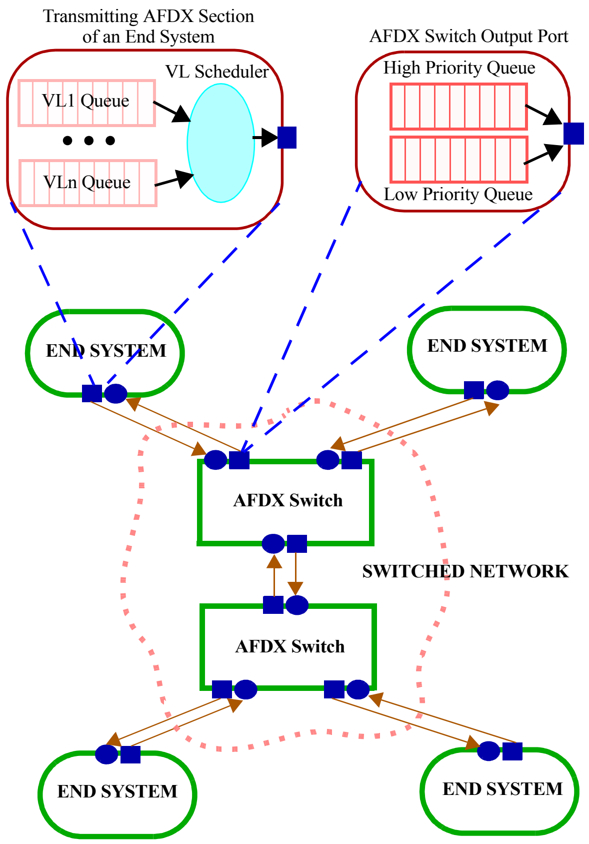

Figure 3. Distributed system based on AFDX

4. Related work

We can find a lot of work devoted to the analysis of end-to-end delays in an AFDX network. The Network Calculus approach has been used in the context of AFDX [14] for obtaining upper bounds to the latency of message flows. The work in [21] proposes a probabilistic analysis of the network, using a stochastic Network Calculus approach, for the computation of the distribution of the delay of a given message flow, but this approach does not guarantee worst-case latencies. The work in [8] presents a model-checking approach based on timed automata to compute the exact worst-case latency of a given message flow, but it is only useful for small examples. The work presented in [1] also shows an algorithm to obtain the exact worst case latencies in AFDX networks by reducing the search space, which is suitable for up to 50 VLs. Finally, a work based on the Trajectory concept [13] is presented in [5] showing that the new method obtains more accurate upper bounds than previous methods, for the worst-case latency in an AFDX network. This work was later extended to use prioritized output queues in the switches [6].

All these works consider the network in isolation, i.e., flows of anonymous packets arrive at the network asynchronously with a minimum interarrival time, and the worst-case latency to transmit the packet is calculated without any relation with the application tasks that have generated the messages. Furthermore, the work in [5] assumes that jitter is null in the sender nodes, which could be considered a restriction for the common case in which a task with variable execution time and preemption effects produces the message. Other recent work about AFDX [4] deals with the use of a component-based design methodology to describe the behavior of the model, and proposes a stochastic abstraction to provide quantitative information on the protocol, that obtains guarantees on latency requirements.

The main difference between our work and the previous ones is that we deal with the calculation of the worst-case latencies in AFDX as a part of the analysis of the end-to-end flow as it is understood by the schedulability analysis theory for distributed systems, i.e., in a holistic way [22][23]. An end-to-end flow is released by a periodic sequence of external events and contains a set of steps, which are tasks executing code in a processor or messages being sent through a communications network. The work in [19] shows that it is possible to compose the response time analysis in the different resources (processors or networks) of a distributed system if the analysis can accept offsets and input jitter for the triggering events and can generate results with best- and worst-case response times.

In this paper we develop a composable response time analysis for the messages in the AFDX network, thus taking into account input offsets and jitter in every message, and producing best and worst-case results that can be composed in a holistic analysis of a distributed system. Messages can be composed by one or more packets. On the contrary, the analyzed previous works only calculate the latencies of each individual packet for the Switched Network depicted in Figure 3. As in [6], our work also considers priorities in the switches.

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 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. We first focus on the analysis with just one switch. Then we extend the analysis to the use of multiple switches.

5.1. Analysis for single-switch systems

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 σ_i being sent through VL_k can be calculated as follows, for the worst case:

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

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 σ_i:

$$Latency = \frac{N_i \cdot 8}{N_{bw}} \tag{5}$$

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:

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

where, Np_i is the amount of bytes corresponding to the packet payload. As can be seen in Eq. (6) when the payload is smaller than 18 bytes the minimum ethernet frame, F_{min}, must be transmitted.

The size of the last packet of a message belonging to stream σ_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:

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

The equation above 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):

$$\begin{aligned} N_{i,\,last} &= O_{Eth} + F_{min} \qquad Np_{i,\,last} \in [1,17] \\ N_{i,\,last} &= O_{Eth} + O_{Prot} + Np_{i,\,last} \qquad Np_{i,\,last} \in [18,1471] \end{aligned} \tag{8}$$

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

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

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

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

5.1.2. Scheduling simple virtual links, L_{VL} (step 1)

For the analysis of a message sent across a simple virtual link, with no sub-VLs, we assume that the technological latency on transmission, L_T, 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 σ_i being sent through VL_k due to the scheduling of the virtual links in a specific processor can be calculated as follows:

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

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

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:

- According to the principles of response-time analysis the analysis technique should be applied for all the message instances that can be in the queue in the worst case busy period [7][10]. 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 packet under analysis contributes with an interference equal to the BAG_k. Notice that the BAG plays the same role as the execution time in traditional response-time analysis, because it represents a time during which the resource, which is the VL scheduler in this case, is unavailable for further transmissions.
- A message in the FIFO queue cannot be preempted, so when calculating the interference of the rest of the messages in the VL, we only need to consider those messages that arrived at the queue before the message under analysis.
- There are no blocking terms as in conventional response time analysis, because these terms account for the delay caused by lower priority activities, but there are no priorities in the VLs.

Since the virtual link uses a FIFO queueing discipline and packets are non-preemptive, we can analyze the latencies using an adaptation of non-preemptive response time analysis [10]. We calculate the worst-case latency of the last packet of the q-th instance of a message from stream σ_i to reach the VL scheduler, as follows:

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

where, $q \cdot p_i - 1$ is the worst-case number of packets contained in the q message instances except the last packet, and $MS(VL_k)$ is the set of message streams that share VL_k with message stream σ_i (excluding itself). The first term in the equation corresponds to the interference of previous packets 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.

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 σ_i that become ready for transmission before the end of the busy period. The number of instances is calculated as indicated in [10]:

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

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 σ_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 σ_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,

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

The interference of the rest of VLs, $I_{VL(ik)}$, can be calculated based on the formula indicated in [2] as follows:

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

where, S_k is the set of VLs in the same processor as VL_k (excluding itself). The following restrictions are defined in the standard

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

These requirements 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 4 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. Table 1 and Table 2 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

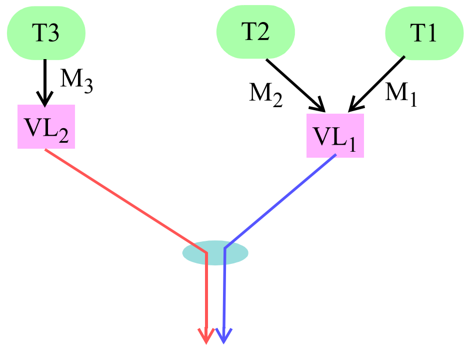

Figure 4. Example with 2 VLs and 3 message streams

TABLE 1. Configuration of VLs

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

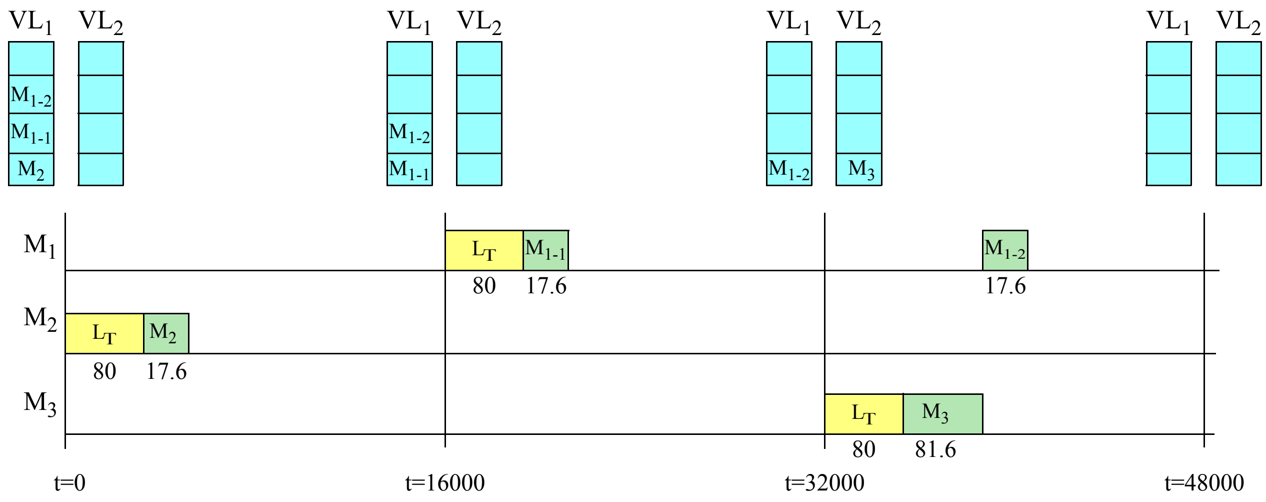

Figure 5. Time Diagram showing the latencies in the VL scheduler, in µs (for visibility, they are not scaled)

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 5 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 L_T=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 $M_{1\text{-}1}$ (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:

- 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}).

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 3.2.4.2 in [2].

5.1.3. Scheduling of sub virtual links, L_{VL} (step 1)

When the VL under analysis has more than one sub-VL, we need to use an alternative analysis from the one in Subsection 5.1.2. The analysis of a message sent across a sub virtual link belonging to a specific virtual link is carried out following similar assumptions to those for simple virtual links. The latency of a message from stream σ_i being sent through sub-VL SVL_{mk} belonging to VL_k due to the scheduling of the VLs in a specific processor can be calculated as:

$$L_{VL(ik)} = L_{SVL(imk)} = L_{SVLQ(imk)} + I_{VL(ik)} \qquad (19)$$

where, $I_{VL(ik)}$ is the worst-case interference from the messages of the other VLs in the same processor generating message stream σ_i, Eq. (17), and $L_{SVLQ(imk)}$ is the worst-case latency for the last packet of a message in the SVL_{mk} queue, including the effects of the messages that can be awaiting in SVL_{mk} and on the other subVLs sharing VL_k.

To obtain the $L_{SVLQ(imk)}$ latency we will create a worst-case scenario in which, when the message under analysis is released, the sub-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 packets sharing the sub-VL, the interference of the previous packets of the message under analysis, and also the interference of the rest of the messages of other sub-VLs sharing the same VL. For this purpose we take into account the following observations based on the ones used in Subsection 5.1.2:

- We analyze the messages in a worst-case busy period.

- Each packet in the sub-VL queue that is ahead of the packet under analysis contributes with an interference equal to the BAG_k.
- Each packet in other sub-VL queues sharing the VL contributes with an interference equal to the BAG_k, up to a bound equal to the number of packets awaiting in SVL_{mk}, according to the round-robin policy used to schedule sub-VLs. Notice that other packets that may be enqueued in the other sub-VLs will be dispatched after the message under analysis, when they get their round-robin turn.
- A message in the sub-VL FIFO queue cannot be preempted, so when calculating the interference of the rest of the messages in the sub-VL, we only need to consider those that arrived at the queue before the message under analysis.

We calculate the interference due to messages in SVL_{mk} for the last packet of the q-th instance of a message from stream σ_i to reach the VL scheduler, as follows:

$$w_{im}(q) = (q \cdot p_i - 1) \cdot BAG_k + \sum_{j \in MS(SVL_{mk})} \left(\left\lfloor \frac{J_j + (q-1) \cdot T_i}{T_j} \right\rfloor + 1 \right) \cdot (p_j \cdot BAG_k) \tag{20}$$

where $q \cdot p_i - 1$ is the worst-case number of packets contained in the q message instances except the last packet, and $MS(SVL_m)$ is the set of message streams that share SVL_{mk} with message stream σ_i (excluding itself). The second term in the equation is the interference by all those messages from other streams that have arrived at the sub-VL queue before the message instance under analysis. The result of this equation, $w_{im}(q)$, is the worst-case latency for the last packet of the *q-th* instance of message stream σ_i to reach the VL scheduler after a critical instant assuming there is only one sub-VL.

Now we have to consider the interference of the rest of messages sent through other sub-VLs sharing VL_k, with the following recurrence, starting with $w_i^0(q) = w_{im}(q)$ and finishing when two consecutive values are equal:

$$w_i^{n+1}(q) = w_{im}(q) + \sum_{l \in N(SVL_{mk})} min(w_{im}(q), I_{lm}^{n+1}(q)) \tag{21}$$

with

$$I_{lm}^{n+1}(q) = \sum_{j \in MS(SVL_{lk})} \left(\left\lfloor \frac{J_j + w_i^n(q)}{T_j} \right\rfloor + 1 \right) \cdot (p_j \cdot BAG_k) \tag{22}$$

where $N(SVL_{mk})$ is the set of sub-VLs that share VL_k with SVL_{mk} excluding itself, and $MS(SVL_{lk})$ is the set of message streams sent through SVL_{lk}. The interference due to messages sent through other sub-VLs, $I_{lm}^{n+1}(q)$, is bounded by $w_{im}(q)$, as the packets enqueued after that time will not influence the last packet of the message stream under analysis according to the round-robin policy.

Eq. (21) 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 σ_i that become ready for transmission before the end of the busy period. The number of instances is calculated as indicated in [10]:

$$Q_i = \left\lceil \frac{J_i + BP_{mk}}{T_i} \right\rceil \tag{23}$$

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

$$BP_{mk}^{n+1} = BP_m^{n+1} + \sum_{l \in N(SVL_{mk})} min(BP_m^{n+1}, BP_l^{n+1}) \tag{24}$$

with

$$BP_m^{n+1} = \sum_{j \in MU(SVL_{mk})} \left\lceil \frac{J_j + BP_{mk}^n}{T_j} \right\rceil \cdot (p_j \cdot BAG_k)$$
$$BP_l^{n+1} = \sum_{j \in MU(SVL_{lk})} \left\lceil \frac{J_j + BP_{mk}^n}{T_j} \right\rceil \cdot (p_j \cdot BAG_k) \tag{25}$$

where, $MU(SVL_{mk})$ is the set of all the message streams using SVL_{mk} including message stream σ_i under analysis in the set of its own VL.

Using the results obtained for the different values of q in Eq. (21), the worst-case latency for the last packet of the q-th instance of message stream σ_i due to the messages that can be waiting on the sub-VLs queues for the same VL can be calculated in the following way:

$$L_{SVLQ(imk)} = \max_{q = 1, 2, \ldots, Q_i} [L_{SVLQ(imk)}(q)] \tag{26}$$

where,

$$L_{SVLQ(imk)}(q) = w_i(q) - (q-1) \cdot T_i \tag{27}$$

5.1.4. 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 queue, which can be considered as the hardware latency provided by the manufacturer,
- and the time spent in the output port queue, due to the interference of the rest of the packets sent to the same destination end system.

Although the output ports of a switch have only two queues, respectively for low and high priorities, selected on a VL basis, we will propose the formulation to consider P priority levels and then the AFDX switch will be a particular case for P=2.

Assuming that the utilization of the output link is under 100%, we can calculate the total latency in the switch for the last packet of message stream σ_i being sent through VL_k as:

$$L_{Sw(ik)} = L_S + L_{SQ(ik)} \tag{28}$$

where, L_S is the hardware latency of the switch and $L_{SQ(ik)}$ is the time waiting in the output port queue, due to the interference of the rest of the packets in that output queue.

To obtain the $L_{SQ(ik)}$ latency we can apply a similar approach as we used for calculating $L_{VLQ(ik)}$, but considering that P priority levels can be present, and using different roles for the response-time analysis. The traditional role of the execution time is now the time taken to transmit each packet in the output link. The periodicity of the packets arriving from a particular VL is the associated BAG.

We will create a worst-case scenario in which, when the packet under analysis is enqueued into the output queue of a specific priority, this queue already contains the worst-case amount of packets that can interfere the transmission. In addition, the higher priority queues will receive the maximum possible traffic during the busy period in which the implied queues are not empty. 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 queues of the same or higher priority as the priority of the VL are not empty. The worst-case busy period is obtained after a critical instant created with the same criteria as in the analysis of $L_{VLQ(ik)}$ or $L_{SVLQ(imk)}$.
- Each packet in the output queue with higher priority or with the same priority and 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.
- A packet in the FIFO queue of a given priority cannot be preempted by packets of the same priority, so when calculating the interference of the rest of the packets of the message under analysis in the output queue, we only need to consider those that arrived at the queue before the packet under analysis.
- A packet of any priority excluding the lowest priority level can be delayed by an amount of time equal to the worst-case transmission time on the physical link of the largest packet with lower priority. This can be considered as a blocking term.

The interference for the q-th packet coming from VL_k to reach the physical output link is given by the following recurrence relation finishing when $w_k^{n+1}(q) = w_k^n(q)$:

$$w_k^{n+1}(q) = w_k^0(q) + \sum_{j \in HP(VL_k)} \left(\left\lfloor \frac{Jp_j + w_k^n(q)}{BAG_j} \right\rfloor + 1 \right) \cdot L_{Trmax(j)} \tag{29}$$

where $HP(VL_k)$ is the set of VLs that have as destination port the outgoing port of VL_k at a higher priority, and $w_k^0(q)$ is an initial value that accounts for all the terms of the interference that are constant for the q-th packet, and is calculated as follows:

$$w_k^0(q) = B_k + (q-1) \cdot L_{Trmax(k)} + \sum_{j \in EP(VL_k)} \left(\left\lfloor \frac{Jp_j + (q-1) \cdot BAG_k}{BAG_j} \right\rfloor + 1 \right) \cdot L_{Trmax(j)} \tag{30}$$

where $EP(VL_k)$ is the set of VLs that have as destination port the outgoing port of VL_k at the same priority, excluding itself. B_k is the blocking term due to packets with lower priority (it will be zero for a lowest priority packet). It can be calculated as:

$$B_k = max(L_{Trmax(j)}) \qquad \forall j \in LP(VL_k) \tag{31}$$

where $LP(VL_k)$ is the set of VLs that have as destination port the outgoing port of VL_k and have a lower priority. The second term in Eq. (30) corresponds to the interference of previous packets of VL_k, and the third term is the interference by all those packets from other VLs at the same priority. The second term of Eq. (29) accounts for the rest of the interference by all those packets from other VLs at a higher priority. 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.

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:

$$Jp_j = J_{Tech} + \sum_{m \in S_j} L_{Trmax(m)} + (L_S - L_S^b) \tag{32}$$

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

Eq. (29) 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 [10]:

$$Q_k = \left\lceil \frac{Jp_k + BP_k}{BAG_k} \right\rceil \tag{33}$$

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 $BP_k^{n+1} = BP_k^n$:.

$$BP_k^{n+1} = B_k + \sum_{j \in DP(VL_k)} \left\lceil \frac{Jp_j + BP_k^n}{BAG_j} \right\rceil \cdot L_{Trmax(j)} \tag{34}$$

where $DP(VL_k)$ is the set of VLs that have as destination port the outgoing port of VL_k at the same or a higher priority, including itself.

Using the results obtained for the different values of q in (29), the worst-case latency for the last packet of the q-th instance of message stream σ_i due to the packets that can be waiting on its associated output queue can be calculated in the following way:

$$L_{SQ(ik)} = \max_{q = 1, 2, \dots, Q_k} [L_{SQ(ik)}(q)] \tag{35}$$

where,

$$L_{SQ(ik)}(q) = w_k(q) - (q-1) \cdot BAG_k \tag{36}$$

5.1.5. Message management at the destination end system, L_{Rec} (step 5)

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

5.1.6. Best-case latencies

In order to calculate the output jitter of the messages sent through the network it is necessary to calculate a lower bound on the 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 σ_i being sent through VL_k can be calculated as:

$$p_i^b = \left\lceil \frac{M_i^b}{Lmax_k - O_{Prot}} \right\rceil \tag{37}$$

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

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

$$Np_{i,last}^b = M_i^b - (p_i^b - 1) \cdot (Lmax_k - O_{Prot}) \tag{38}$$

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):

$$\begin{array}{ll} N_{i,last}^b = O_{Eth} + N_{min} & Np_{i,last}^b \in [1,17] \\ N_{i,last}^b = O_{Eth} + O_{Prot} + Np_{i,last}^b & Np_{i,last}^b \in [18,1471] \end{array} \tag{39}$$

And then we can calculate the best-case latency of a last packet transmitted through the Ethernet link applying Eq.

(5) with this size:

$$L^b_{Tr(i)} = \frac{N^b_{i,last} \cdot 8}{N_{bw}} \tag{40}$$

Step 1: Scheduling of virtual links

The latency of a message from stream σ_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 $I^b_{VL(ik)}$, which is the best-case interference from the messages of the other VLs in the same processor generating message stream σ_i:

$$L^b_{VL(ik)} = L^b_{VLQ(ik)} + I^b_{VL(ik)} = (p^b_i - 1) \cdot BAG_k + L_{Tmin} \tag{41}$$

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 $L^b_{Rec} = L^b_R$.

5.1.7. Total latency

The worst-case latency, L_{ik}, for a message stream σ_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:

$$L_{ik} = L_{VL(ik)} + 2 \cdot L_{Tr(i)} + L_{Sw(ik)} + L_{Rec} + J_i \tag{42}$$

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

$$L^b_{ik} = [(p^b_i - 1) \cdot BAG_k + L_{Tmin}] + 2 \cdot L^b_{Tr(i)} + L^b_S + L^b_{Rec} \tag{43}$$

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

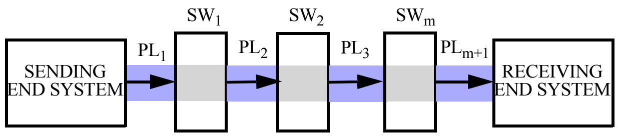

Figure 6. Communication process with *m* switches

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 6 shows the communication process with multiple switches. In this case, the worst-case latency for the message stream σ_i being sent through VL_k and crossing *m* switches can be calculated as follows:

$$L_{ik} = L_{VL(ik)} + \sum_{\forall link(i)} L_{Tr(i)} + \sum_{\forall switch(i)} L_{Sw(ik)} + L_{Rec} + J_i \tag{44}$$

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

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

$$L^b_{ik} = [(p^b_i - 1) \cdot BAG_k + L_{Tmin}] + \sum_{\forall link(i)} L^b_{Tr(i)} + \sum_{\forall switch(i)} L^b_S + L^b_{Rec} \tag{45}$$

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

To analyze a distributed application it is necessary to integrate the analysis in the processors and in the network. Several response-time analysis techniques exist for the processors [12][17][18][22][23] and they can be combined with the developed response time analysis for AFDX networks using the composition approach presented in [19].

One of the interesting properties of 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 σ_i shown in Figure 7, sent at the finalization of task τ_{aj-1} and activating, in turn, τ_{aj+1} in its end-to-end flow Γ_a. Task τ_{aj} is just the model of the σ_i message in the end-to-end flow.

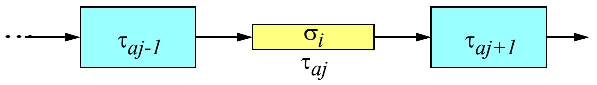

Figure 7. Portion of an end-to-end flow with a message stream sent through an AFDX network

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 τ_{aj-1} :

$$J_i = R_{ij-1} - R^b_{ij-1} \qquad (46)$$

The worst and best-case latencies of the AFDX message shown in equations (44) and (45) already take into account this jitter, and are relative to the best possible release time, which is R^b_{ij-1}. Therefore, the worst- and best-case response times of σ_i (or τ_{aj}) are obtained as:

$$\begin{aligned} R_{ij} &= R^b_{ij-1} + L_{ik} \\ R^b_{ij} &= R^b_{ij-1} + L^b_{ik} \end{aligned} \qquad (47)$$

where k is the index of the VL through which message stream σ_i is sent. From these values we would calculate inherited offsets and jitters that can be used in the heterogeneous response-time analysis algorithms [19].

7. Case study and validation

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. In order to validate the holistic schedulability analysis technique proposed, we discuss the results that this technique is able to obtain, compared to the related work.

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

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 is 80 µs, L_{Tmin}=J_{Tech}= 40 µs., $L_S = 100\mu s$, $L^b_S = 70\mu s$, L^b_{Rec}=40 µs.

7.1. Situation 1

In this situation (Figure 8), 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 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

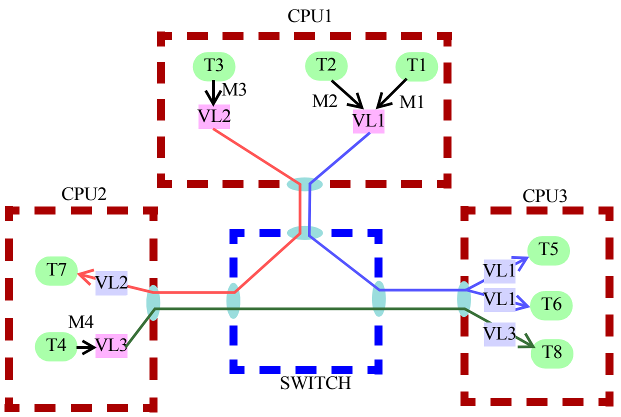

Figure 8. Diagram for Situation 1

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 8 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 task T5, which is receiving this message (this jitter is 16.2132 milliseconds).

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

7.2. Situation 2

In this situation (see Figure 9 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 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

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.

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

In this case the contribution of message M1 to the output jitter, i.e., the difference between the worst 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.

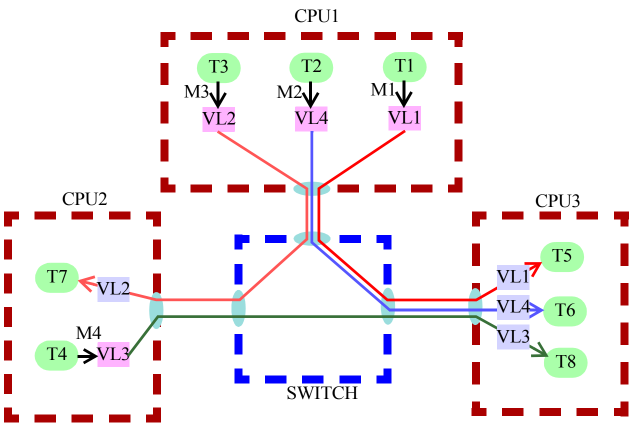

Figure 9. Diagram for Situation 2

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 now has 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.

7.3. Highlights of the proposed technique

The analysis technique proposed in this work is an adaptation of the holistic analysis initially developed by Tindell and Clark [23] and validated by Palencia et al [16]. We have added the analysis of the round-robin VL scheduler handling the FIFO queues of non-preemptive packets at the VLs, and the analysis of priority queues of non-pre-emptive packets at the output ports of a switch. It is well known that this analysis is pessimistic (it obtains a safe upper bound of the response times), and that part of this pessimism can be eliminated by applying offset-based techniques [17] and also by adding the analysis of best-case response times, thus reducing the jitter. So the purpose of this section is not to show the results of our technique over extensive examples, as it is expected to obtain similar performance than the holistic technique in which it is based, Our objective is to highlight that our analysis is useful to analyze the overall system, even if it can be pessimistic.

In order to compare our algorithm with existing techniques, we need to take into account the limitations of those techniques. We can only compare partially, i.e., for the analysis of the Switched Network in Figure 3, and only for single packets. We have selected the example in Figure 3 in [5], showing a network composed by 5 VLs and 3 switches. Table II in [5] shows the latencies of single packets obtained by the following techniques introduced in the related work: the exact worst-case (model checking approach), the basic Network Calculus (NC), the Network Calculus with grouping (NCG), the basic Trajectory approach (BT), and the optimized Trajectory approach (OT). The last one obtains the exact worst-case latencies for this example while the others obtain upper bounds. Table 10 reproduces Table II in [5] with the results of the holistic approach for this example. Our analysis technique obtains slightly better results than the basic Network Calculus and is a little bit more pessimistic than the others.

The OT approach presented in [6] has an example similar to the previous one, where VL v1 has a high priority while the other VLs have low priority. In this example, for the analysis of v1 our algorithm obtains the exact worst-case response time (232 μs), as happens with the OT approach. The other VLs have the same results as in Table 10.

TABLE 10.worst-case latencies in μs

VL	NC	NCG	BT	OT	Holis tic
v1	313.2	273.6	312	272	312
v2	192.4	192.4	192	192	192
v3	313.2	273.6	272	272	312
v4	313.2	273.6	272	272	312
v5	217.2	177.6	216	176	216

On the other hand, as we introduced in Section 4, model checking [8] is able to compute the exact worst-case latency of a packet in a VL, but at a very high computational cost. The approach in [1] uses model checking with a reduced search space to obtain the exact worst-case latencies in reasonable times for up to 50 VLs. We have applied our holistic approach to the switched network proposed in Figure 13 in [1], which consists of 4 switches and 58 VLs. We have analyzed this example for different practical loads (after 45% average utilization of the output ports of the switches the network is almost unresponsive) and configurations of VLs (crossing up to three switches), and the maximum time spent by our algorithm is 0.02 seconds. Table II in [1] shows that for a similar example with 64 VLs, the calculation of exact latencies takes more than one hour to enhance by 9% the results obtained by Network Calculus. As the authors of [1] state, that technique is not applicable yet for real industrial configurations with more than 1000 VLs. We have applied our algorithm over an example of this size consisting of 16 switches and 1000 VLs (crossing up to 6 switches). Figure 10 shows the maximum and average execution times taken by our algorithm to analyze the example with different utilizations of the output ports of the switches. We can see that all the execution times are reasonable enough to allow design space exploration. All the tests have been made on an Intel Core i7 CPU 860 at 2.93 GHz without exploiting its parallelism.

In summary, the key points in favor of the holistic analysis technique that we propose for distributed systems based on AFDX networks are:

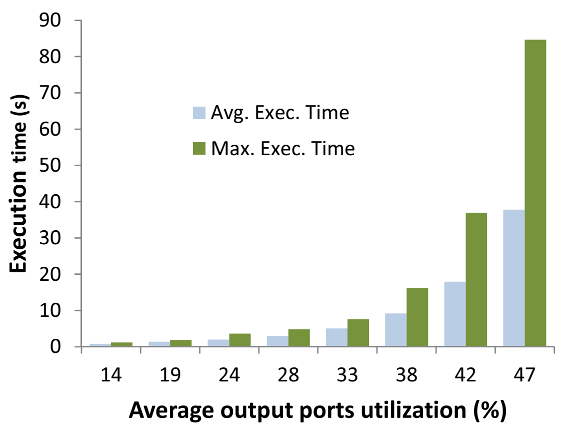

Figure 10. Execution times for 1000 VLs

- It deals with the whole system and can be integrated with other techniques for the processors based on the composition approach presented in [19].
- It allows the tasks to send messages composed of one or more packets, taking into account the contention in the VL scheduler and also input and output jitter.
- Although the technique is pessimistic compared to other existing approaches, it is scalable and it can be apply to real systems.
- As our analysis is made in several steps, following the composition approach in [19], the analysis of the switched network (*Step 4*) can be replaced by more precise future techniques (e.g., adding offsets). As future work, we could also evaluate if existing less pessimistic techniques, e.g. optimized Trajectory approach, can be adapted to this holistic analysis.

8. Conclusions and future work

In this paper we have developed a new response-time analysis technique for AFDX networks. This analysis technique can be combined with other response-time analysis techniques to analyze heterogeneous 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 [15]. 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.

Another planned extension is the adaptation of offset-based analysis techniques to analyze synchronized message streams in the AFDX network.

References

[1] M. Adnan, J.-L. Scharbarg, and C. Fraboul, “Minimizing the Search Space for Computing Exact Worst-Case Delays of AFDX Periodic Flows,” in Proc. of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES’11), Västerås, Sweden, pp. 294-301, 2011.

[2] Airlines Electronic Engineering Committee, Aeronautical Radio INC., “ARINC Specification 664P7: Aircraft Data Network, Part 7 - Avionics Full Duplex Switched Ethernet (AFDX) Network,” June 27, 2005.

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

[4] A. Basu, S. Bensalem, M.Bozga, B. Delahaye, A. Legay, and E. Sifakis, “Verification of an AFDX Infrastructure using Simulations and Probabilities,” in Proc. of the First international conference on Runtime verification, RV 2010, Malta, LNCS 6418, pp. 330-344, 2010.

[5] H. Bauer, J.-L. Scharbarg, and C. Fraboul, “Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach,” IEEE Transactions on Industrial Informatics 5(4), pp. 521-533, 2010.

[6] H. Bauer, J.-L. Scharbarg, and C. Fraboul, “Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources,” Real-Time Systems Journal 48, pp. 101-133, 2012.

[7] R.J. Bril, J.J. Lukkien, and W.F.J. Verhaegh, “Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption,” Real-Time Systems Journal 42(1-3): 63-119, 2009.

[8] H. Charara, J.-L. Scharbarg, J. Ermont, and C.Fraboul, “Methods for bounding end-to-end delays on an AFDX network,” in Proc. of the 18th Euromicro Conference on Real-Time Systems, Dresden, Germany, pp. 193-202, 2006.

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

[10] 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.

[11] M. González Harbour, J.J. Gutiérrez, J.C. Palencia, and J.M. Drake, “MAST: Modeling and Analysis Suite for Real Time Applications,” in Proc. of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, pp. 125-134, 2001.

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

[13] S. Martin and P. Minet, “Schedulability analysis of flows scheduled with FIFO: Application to the expedited forwarding class,” in Proc. of the 20th Int. Parallel and Distributed Processing Symposium, Rhodes Island, Greece, 2006.

[14] F. Frances, C. Fraboul, and J. Grieu, “Using network calculus to optimize the AFDX network,” in Proc. of the ERTS, Toulouse, France, 2006.

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

[16] J. C. Palencia, J.J. Gutiérrez, and M. González Harbour, “On the Schedulability Analysis for Distributed Hard Real-Time Systems,” in Proc. of the 9th Euromicro Workshop on Real-Time Systems, Toledo, Spain, pp. 136-143, 1997.

[17] J.C. Palencia, and M. González Harbour, “Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems,” in Proc. of the 20th IEEE Real-Time Systems Symposium, Phoenix, USA, pp. 328-339, 1999.

[18] J.C. Palencia and M. González Harbour, “Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF,” in Proc. of the 15th Euromicro Conference on Real-Time Systems, Porto, Portugal, pp. 3-12, 2003.

[19] J. M. Rivas, J.J. Gutiérrez, J.C. Palencia, and M. González Harbour, “Schedulability Analysis and Optimization of Heterogeneous EDF and FP Distributed Real-Time Systems,” in Proc. of the 23rd Euromicro Conference on Real-Time Systems, Porto, Portugal, pp. 195-204, 2011.

[20] 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`

[21] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, “A probabilistic analysis of end-to-end delays on an AFDX network,” IEEE Transactions on Industrial Informatics 5(1), pp. 38-49, 2009.

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

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