Verification of an AFDX infrastructure using simulation and probabilities

Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Delahaye, Axel Legay, Emmanuel Sifakis

▶ **To cite this version:**

Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Delahaye, Axel Legay, et al.. Verification of an AFDX infrastructure using simulation and probabilities. Runtime Verification - First International Conference, RV 2010, Nov 2010, St. Julians, Malta. pp.330-344, 10.1007/978-3-642-16612-9 . hal-00557717

HAL Id: hal-00557717

https://hal.archives-ouvertes.fr/hal-00557717

Submitted on 19 Jan 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Verification of an AFDX Infrastructure using Simulations and Probabilities*

Ananda Basu[1], Saddek Bensalem[1], Marius Bozga[1], Benoit Delahaye[2], Axel Legay[2], and Emmanuel Sifakis[1]

[1] Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS
[2] INRIA/IRISA, Rennes, France

Abstract. Until recently, there was not a strong need for networking inside aircrafts. Indeed, the communications were mainly cabled and handled by etherned protocols. The evolution of avionics embedded systems and the number of integrated functions in civilian aircrafts has changed the situation. Indeed, those functionalities implies a huge increase in the quantity of data exchanged and thus in the number of connections between functions. Among the available mechanisms provided to handle this new complexity, one find Avionics Full Duplex Switched Ethernet (AFDX), a protocol that allows to simulate a point-to-point network between a source and one or more destinations. The core idea in AFDX is the one of Virtual Links (VL) that are used to simulate point-to-point communication between devices. One of the main challenge is to show that the total delivery time for packets on VL is bounded by some predefined value. This is a difficult problem that also requires to provide a formal, but quite evolutive, model of the AFDX network. In this paper, we propose to use a component-based design methodology to describe the behavior of the model. We then propose a stochastic abstraction that allows not only to simplify the complexity of the verification process but also to provide quantitative information on the protocol.

1 Introduction

Until recently, there was not a strong need for networking inside aircrafts. Digital technologies were initially introduced at the control platform of the aircrafts with the fly-by-wire technologies.

The evolution of avionics embedded systems and the number of integrated functions in civilian aircrafts implied a huge increase in the quantity of data exchanged and thus in the number of connections between functions. The Aircraft Data Networks used until now had either point to point connections, which incurred a high cost in aircraft production as well as increase of weight, or mono transmitter buses with very low performances (100Kbits/s). The innovation of *Avionics Full Duplex Switched Ethernet (AFDX)* [2] was to use an open standard such as Ethernet and take advantage of its high bandwidth, 100Mbps, and

* This work has been supported by the Combest EU project.

the use of cheap COTS components. AFDX also offers the capability to easily extend the avionic network with new devices, as well as to reduce the wiring.

For a network to be suitable for use in critical applications, it must be reliable and deterministic. In AFDX reliability is achieved with redundancy while determinism with the definition of Virtual Links (VL), which put constraints on the allowed traffic. A network is deterministic if we can guarantee an upper bound for the time a message needs to be delivered to its destination. For AFDX such upper bounds can be provided with analytical methods [10]. The bounds obtained are over approximations of the worst case and the analysis can only be performed on very abstract models [8]. There is thus the need for new methods that will guarantee more realistic upper bounds on more realistic models.

In a very recent work [6], we suggested *stochastic abstraction*. This technique can ease the verification process of large heterogeneous systems by abstracting some of the components of the system with probability distributions. The ability to add stochastic information can also be used to compute a probability for the system to satisfy the property. The latter can efficiently be done with *Statistical Model Checking* (SMC) [12, 19, 21] that has recently been proposed as an alternative to avoid an exhaustive exploration of the state-space of the model. The core idea of SMC is to conduct some simulations of the system and then use results from the statistic area in order to decide whether the system satisfies the property. Statistical model checking techniques can also be used to estimate the probability with which a system satisfies a given property [12, 11]. Of course, in contrast with an exhaustive approach, a simulation-based solution does not guarantee a correct result. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory and time intensive than exhaustive ones, and are sometimes the only option [22, 14]. Statistical model checking gets widely accepted in various research areas such as systems biology [9, 15] or software engineering, in particular for industrial applications. The technique in [6], which combines statistical model checking and stochastic abstraction, was capable of verifying properties of a model with more than 2^{3000} states in a few minutes.

In this experimental paper, we propose to apply the stochastic abstraction principle on AFDX networks. Our contributions are twofolds:

1. **Model of the network.** In this paper, we propose a BIP model for AFDX architecture. BIP [5] is a tool for specifying components and component interactions. One of the very attractive features of BIP is its ability to generate executions of composite systems. This is required to compute the stochastic abstraction as well as to apply statistical model checking. Another advantage of BIP is that it permits to give a very detailed description of each component. Also, components can be developed by independent teams, which decrease the complexity of the design.
2. **Verification.** We then examine the *latency requirements* property in AFDX, i.e., we check that the total delivery time for packets on virtual links is smaller than some predefined values. The difficulty is that our model of AFDX is constituted of many BIP components – this is needed to obtain an accurate

model of the network. Combining these components lead to a system that is too big (in terms of states) to be analyzed by classical verification techniques such as model checking. In order to overcome the problem, we suggest to abstract some of these components with probability distributions, hence producing another BIP model of the network that is a stochastic abstraction of the original one. We then apply statistical model checking to estimate a value of the bound for which the requirement is satisfied with probability 1. This is an important feature as correct upper bounds are mandatory for certification. We also show that one can use our approach to compute the probability that the latency requirement is satisfied for a given value of the bound. This latest feature is of interest to adapt/reconfigure the network for better average performances.

We believe that our work is interesting as it proposes (1) a very detailed and accurate model of an AFDX network that is obtained by a component-based design methodology (this also illustrate the advantages of the BIP toolset), and (2) an efficient formal technique for verifying properties and provide quantitative informations on this model.

Structure of the paper. Section 2 introduces BIP and SMC while Section 3 is dedicated to AFDX. In Section 4, we propose our compositional design methodology to generate formal models of AFDX networks. Section 5 focuses on a stochastic abstraction and presents our experiments. Finally, Section 6 concludes the paper and discuss related works.

2 Preliminaries

In this section, we briefly introduce the BIP framework that is used to define components and component interactions. BIP has the ability to simulate execution of composite systems. We then introduce statistical model checking that is a technique used to estimate/or validate the probability for a stochastic system to satisfy some given property. Statistical model checking requires the ability to simulate the model, which is exactly what BIP can provide.

2.1 The BIP Framework

The BIP framework, introduced in [5], supports a methodology for building systems from *atomic components*. It uses *connectors*, to specify possible interaction patterns between components, and *priorities*, to select amongst possible interactions. In BIP, data and their transformations can be written directly in C. Atomic components are finite-state automata extended with variables and ports. Ports are action names, and may be associated with variables. They are used for synchronization with other components. Control states denote locations at which the components await for synchronization. Variables are used to store local data. Composite components allow defining new components from sub-components (atomic or composite). Components are connected through flat or hierarchical *connectors*, which relate ports from different sub-components. Connectors represent sets of *interactions*, that are, non-empty sets of ports that have to be jointly executed. They also specify guards and transfer functions for each interaction, that is, the enabling condition and the exchange of data across the ports of the interacting components. *Priorities* are used to select amongst simultaneously enabled interactions. They are a set of rules, each consisting of an ordered pair of interactions associated with a condition. When the condition holds and both interactions of the corresponding pair are enabled, only the one with higher-priority can be executed.

BIP is supported by an extensible tool-set which includes functional validation, model transformation and code generation features. Actually, code generation targets both simulation and implementation models (e.g., distributed, multi-threaded, real-time, etc.). In particular, simulation is driven by a specific middleware, *the BIP engine*, which allows to generate, explore and inspect execution traces corresponding to BIP models.

2.2 Statistical Model Checking

Consider a stochastic system $\mathcal{S}$ and a property ϕ. *Statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions : (1) **Qualitative :** Is the probability for $\mathcal{S}$ to satisfy ϕ greater or equal to a certain threshold θ? and (2) **Quantitative :** What is the probability for $\mathcal{S}$ to satisfy ϕ? Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise.

Qualitative Answer using Statistical Model Checking The main approaches [21, 19] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\phi)$, to determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β). A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [21] for details). A solution is to use an *indifference region* $[p_1, p_0]$ (with θ in $[p_1, p_0]$) and to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. We now very briefly sketch an hypothesis testing algorithm that is called the *sequential probability ratio test (SPRT in short)* [20]. Our intention is not to give too much details on SPRT – the interested reader will read [20].

In SPRT, one has to choose two values A and B ($A > B$) that ensure that the strength (α, β) of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1[20]. This has the advantage of minimizing the number of simulations. In his thesis [21], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure. When one has to test $\theta \geq 1$ or $\theta \geq 0$, it is better to use *Single Sampling Plan* (SSP) (see [21, 17, 19] for details) that is another hypothesis testing algorithm whose number of simulations is pre-computed in advance. In general, this number is higher than the one needed by SPRT, but is it known to be optimal for the above mentioned values. More details about hypothesis testing algorithms and a comparison between SSP and SPRT can be found in [17].

Quantitative Answer using Statistical Model Checking In [12, 16] Peyronnet et al. propose an estimation procedure to compute the probability p for $\mathcal{S}$ to satisfy ϕ. Given a *precision* δ, Peyronnet's procedure, which we call PESTIMATION, computes a value for p' such that $|p' - p| \leq \delta$ with *confidence* $1 - \alpha$. The procedure is based on the *Chernoff-Hoeffding bound* [13]. Let $B_1 \dots B_m$ be m discrete random variables with a Bernoulli distribution of parameter p associated with m simulations of the system. Recall that the outcome for each of the B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. Let $p' = (\sum_{i=1}^{m} b_i)/m$, then Chernoff-Hoeffding bound [13] gives $Pr(|p' - p| > \delta) < 2e^{-\frac{m\delta^2}{4}}$. As a consequence, if we take $m \geq \frac{4}{\delta^2} \log(\frac{2}{\alpha})$, then $Pr(|p' - p| \leq \delta) \geq 1 - \alpha$. Observe that if the value p' returned by PESTIMATION is such that $p' \geq \theta - \delta$, then $\mathcal{S} \models Pr_{\geq \theta}$ with confidence $1 - \alpha$.

Playing with Statistical Model Checking Algorithms The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer. This number may change from executions to executions and can only be estimated (see [21] for an explanation). However, some generalities are known. For the qualitative case, it is known that, except for some situations, SPRT is always faster than SSP. PESTIMATION can also be used to solve the qualitative problem, but it is always slower than SSP [21]. If θ is unknown, then a good strategy is to estimate it using PESTIMATION with a low confidence and then validate the result with SPRT and a strong confidence.

3 The Avionics Full Duplex Switched Ethernet

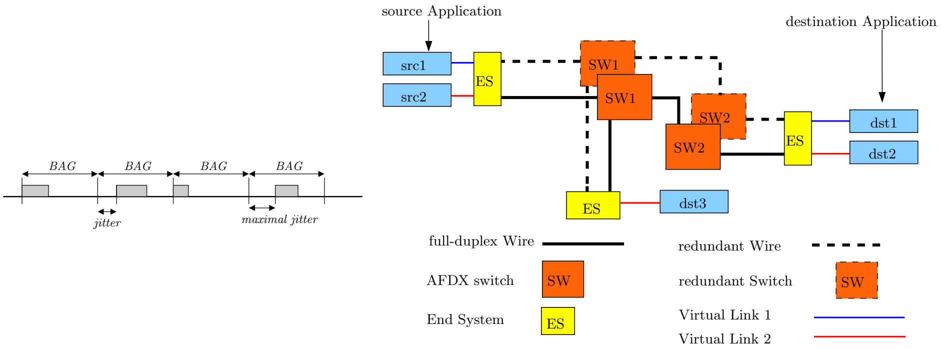
AFDX [2] is a standard developed by Airbus for building highly reliable, time deterministic aircraft data networks (ADNs) based on commercial, off-the shelf Ethernet technology.

The first standard defined for ADNs has been ARINC 429 [1]. This standard, developed over thirty years ago and still widely used today, has proven to be highly reliable in safety critical applications. It relies on point-to-point unidirectional bus with single transmitter and up to twenty receivers. Consequently, ARINC 429 networks need a significant amount of wiring and imply a non-negligible aircraft weight increase.

ARINC 664 has been defined as the next-generation ADNs standard. It is based upon IEEE 802.3 Ethernet and uses commercial off-the-shelf hardware thereby reducing costs and development time. AFDX is formally defined in Part 7 of the ARINC 664 specification [2]. It has been developed by Airbus Industries for the A380 and since then, it has been accepted by Boeing and used on the Boeing 787 Dreamliner. AFDX bridges the gap on reliability of guaranteed bandwidth from the original ARINC 664 standard. It features a star topology network of up to 24 end-systems connected to a switch, where each switch can be bridged together to other switches on the network. Based on this topology, AFDX is able to significantly reduce wire runs thus reducing overall aircraft weight. Additionally, AFDX provides dual link redundancy and Quality of Service (QoS).

Virtual Links. AFDX offer to avionics engineers the capability to think in terms of point-to-point connections of their applications at design time, just as they did in the past with ARINC 429. This capability is provided through the notion of *virtual links* (VL), that are, logical unidirectional connections from one transmitter end-system to one or many receiver end-systems. Moreover, virtual links are used to define and control the QoS within AFDX networks. They are annotated with non-functional characteristics including (i) the *bandwidth allocation gap* (BAG), the time interval allocated for the transmission of one packet, (ii) the *minimum* and *maximum packet size* and (iii) the *jitter* allowed for transmission, with respect to the beginning of the BAG. These characteristics can be visualized in figure 1a.

End Systems. The end-systems are the entry points on AFDX networks. They realize the interface between the application software and the network infrastructure. They mainly perform three tasks: traffic regulation, scheduling and redundancy management. First, *traffic regulation* transforms arbitrary (not regulated) flows of packets sent by applications such that they meet their corresponding virtual links characteristics. More precisely, stores packets and delivers them with the correct rate i.e., exactly one packet per BAG, without jitter. Second, *scheduling* organize the global flow obtained from several virtual links before their delivery on the channel. Packets coming from the traffic regulator and corresponding to different virtual links are interleaved in order to ensure



(a) QoS Characteristics of Virtual Links

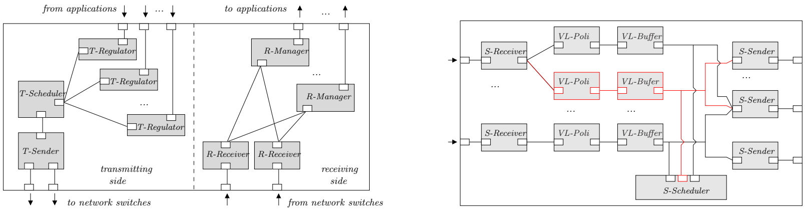
(b) Overview of AFDX networks.

Fig. 1: Details of AFDX.

QoS characteristics such as bounded delivery time or bounded jitter. Third, *redundancy management* enforces dual redundancy. On transmission side, packets are indexed, duplicated and sent towards destination on two distinct channels. On reception side, a *first valid wins* policy is applied, that is, the first valid packet received is delivered to the destination application, whereas the second is silently discarded.

Switches. AFDX switches are the core elements of AFDX networks. They perform tasks such as frame filtering, traffic policing, switching, and monitoring. *Frame filtering* discards invalid packets according to various integrity rules (concerning packet size, sequence numbers, incoming path, etc). In a similar way, *traffic policing* maintains the traffic for each virtual link within its (statically) declared characteristics i.e., avoid fault propagation such as network flooding because of faulty end-systems or switches. The *switching* functionality performs the routing of packets towards their destination(s). The routes are statically defined for every virtual link. Finally, *monitoring* realizes various logging and other administrative operations.

Requirements. The use of AFDX in safety-critical systems is usually subject to extra requirements. In particular, there are *latency requirements*, that are, the total delivery time for packets on virtual links must be smaller than some predefined values. For example, such requirements are mandatory when AFDX is used to transport data needed for navigation and control applications running on board. They usually have to be formally verified as part of the certification process. Nevertheless, their verification is difficult since they are system-level properties depending both on the network topology (physical infrastructure) and on the whole set of virtual links (application traffic) deployed on it.



(a) Model of AFDX End-Systems

(b) Model of AFDX Switches

Fig. 2: Details of the modeling of AFDX in BIP.

4 A BIP Model for AFDX Systems

We have developed a systematic way to construct accurate functional models of AFDX networks in BIP. Network models are structural, that is, they are interconnected assemblies of models of AFDX entities (end-systems and switches), following the physical connections and reflecting the static deployment on virtual links. This construction arises naturally given the BIP modeling principles and enforces a clear separation between functional (behavior) and architectural (connection) elements. Moreover, it allows the development of models for real-sized AFDX networks, of arbitrary complexity with no difficulty.

The BIP components modeling AFDX end-systems and switches are parametric and can accommodate arbitrary but statically fixed numbers of virtual links. Their inner architecture reflects the functional decomposition established by the AFDX standard [2]. For example, the BIP model for end-systems is composed of traffic regulator(s), scheduler, sender, receiver(s) and redundancy manager(s) sub-components, inter-connected as shown in figure 2a.
All sub-components are atomic and have been explicitly modeled using discretized timed automata. As an example, we provide the model of the atomic traffic regulator component in figure 3. The BIP component representing AFDX switches has been constructed in a similar way. Its inner architecture is shown in figure 2b. The switch model includes atomic sub-components implementing traffic classification, traffic policing, in packet buffering for each virtual link, scheduling and finally, out packet buffering, for each outgoing connection.

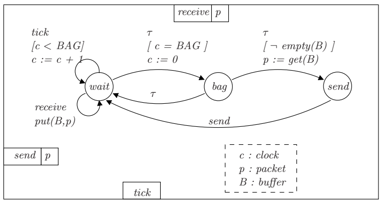


Fig. 3: Model of Traffic Regulator

In particular, let us remark that the AFDX standard leaves open the scheduling algorithms to be used within end-systems and switches. Nevertheless, our BIP models provide concrete implementations, that is, round-robin on incoming virtual links. We made this choice since, while being very easy to implement, it introduces relatively low jitter for multiplexing regulated flows of packets incoming on several virtual links. Still, the model can be easily changed to use other scheduling policy, if needed. We have identified nine types of atomic components in BIP necessary to model any AFDX system (excluding the application atomic components). As an example, a switch model with two input (each having ten Virtual Links) and one output, contains twenty-four atomic components. An end system generating ten Virtual Links is modeled by twenty-two atomic components.

BIP components interact using two categories of connectors. First, there are *data* connectors, as illustrated in figures 2a or 2b, which transport (abstractions of) data packets between various components. These connectors have an unique *sender* port and one or many *receiver* ports. At any interaction an abstract packet (a data structure containing the virtual link id, size, sequence numbers, etc.) is transferred simultaneously from the sender to all the receivers. Second, there is a global *tick* connector (not illustrated) which realizes the time synchronization within the whole model. This connector synchronizes all the *tick* ports, occuring within all the atomic components. Whenever a tick interaction takes place, it correspond to a discrete progress of time by one time unit. Since all components participate, the progress of time is therefore *synchronously* observed/followed by all of them. The time step has been chosen as 1 μs (microsecond) in order to correspond to the magnitude for sending frames on 100Mbps Ethernet networks. That is, the transmission duration for an Ethernet frame takes between 5 and 117 time units (i.e., microseconds).

5 Verification Methodology and Experiments

In this section, we present the experiments we conducted using our BIP model of the AFDX infrastructure. We are interested in verifying the latency requirement property introduced in Section 3. The problem with our model is that, as we have seen, switches are complex components, providing multiples functionalities. This complexity, which is needed to obtain a realistic model of the system, may prevent the use of classical model checking algorithms (state-space explosion). This is especially the case for AFDX systems made of many (e.g., tens) switches and supporting many (e.g., hundreds or thousands) virtual links. However, in order to reason on the latency requirement, we are only interested in the time needed for a message to go through the switch. We suggest to exploit this observation in order to reduce the complexity of the verification process.

More precisely, our idea (which is similar to the one we recently introduced in [6]) is to abstract switch complexity by replacing the switches with probability distributions on the delays for a message to cross them. We thus obtain a *stochastic abstraction* of our AFDX model. We can then use statistical model checking techniques to verify the end-to-end delay properties.

As a running example, let us consider the AFDX network given in Figure 5a. This network is constituted of three switches, five source end-systems, and one destination end system. The source end-systems are connected to the destination end system with several virtual links. For such a network with ten Virtual Links per end-system, the BIP model contains 213 atomic components. As we shall see in the next section, the number of virtual links going out from an end-system and the size of the BAGs may vary from experiments to experiments.

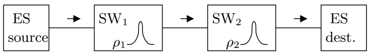


Fig. 4: Abstract stochastic model obtained for a designated virtual link VL_0

Our idea is to replace each switch by several probability distributions, one for each Virtual link. The result will be a BIP model in where the switches are replaced by several automata generating the distribution; an example is given in Figure 4. One can again simulate this model and then use statistical model checking algorithms to verify its properties. As we shall see, these automata can directly be encoded in the BIP engine, which simplify the structure of the model.

We first give details on how to compute the probability distributions and then we discuss the experimental results for several scenarios, each of them depending on the number of virtual links and size of BAGs.

5.1 Stochastic Abstraction

In this section, we briefly describe our approach to learn the probability distributions that abstract switches behaviors.

For each switch and each virtual link, we try to estimate a probability distribution on the delay for a packet to cross the switch. This is done by running the BIP model corresponding to architecture of Figure 5a. For example, let assume that we made 33 measures on a given switch, i.e. the virtual links has sent 33 packets through the switch. The result will be a series of delay values and, for each value, the number of times it has been observed. As an example, delay 5 has been observed 3 times, delay 19 has been observed 30 times. The probability distribution is represented with a table of 33 cells. In our case, 3 cells of the table will contains the value 5 and 30 will contain the value 19. The BIP engine will select a value in the table in a uniform way.

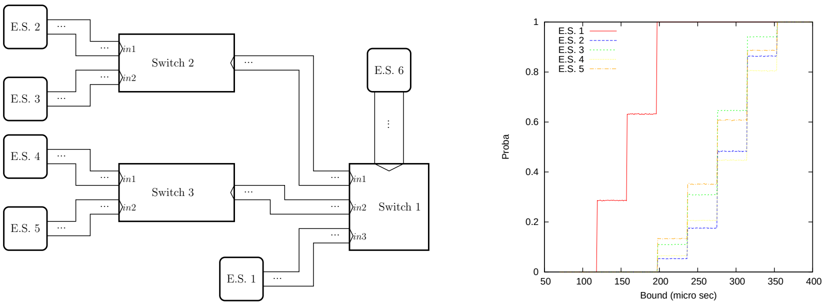
VL's per End System	Switch	Time
10	2	0:08:49.85
	1	0:36:14.06
20	2	0:19:24.17
	1	1:34:36.08
30	2	0:31:57.96
	1	2:03:22.70

Table 1: Simulation times in seconds

According to our observation, 1000 simulations are enough to obtain an accurate estimation of the probability distribution. However, for confidence reasons, we have conducted 4000 experiments. Figure 1 reports on the time needed to conduct these simulations switch by switch.

5.2 Experiments

We now report on our experiments. We are mainly interested in estimating a bound on the total delivery time for packets on virtual links. We are also



(a) Architecture of the experiments

(b) Results for Scenario 1

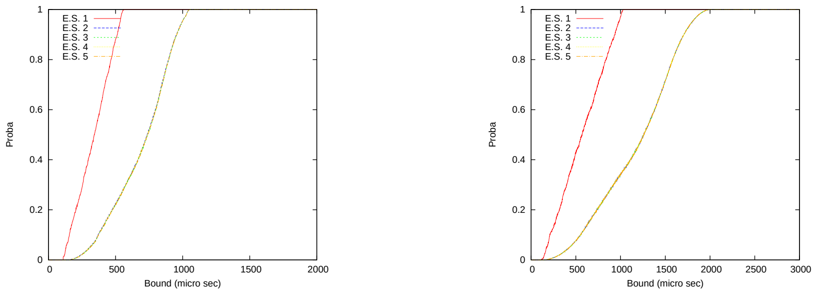
Fig. 5: Architecture of the Experiments and results for Scenario 1

interested in computing the probability that the total delivery time for packets is smaller than a given bound. This bound may be a requirement provided by the user. Verifying these properties is only illistrations of what we can do with our stochastic abstraction.

We consider the AFDX architecture given in Figure 5a, but our methodology applies to any AFDX architecture. We assume that the switches are replaced by probability distributions computed as explained in Section 5.1. As we have seen in the previous section, the number of virtual links and the size of the BAGs and the frames may vary. It is important to study the influence of these variation on the latency requirement. This will be done with the following scenarios.

Scenario 1 In this scenario, each source end-system is connected to the destination end-system with a single virtual link. Each of these five virtual links have the same parameters, that are BAG=4 ms, frame size=500 Bytes. Our first experiment was to compute, for each link, the probability that the total delivery time for packets is smaller than a given bound, until we reach probability one. We were using the ESTIMATION algorithm with a precision of of 0.01 and a confidence of 0.01 to estimate probabilities for bounds between 1 and 400 micro seconds. The results, which were obtained in less than 2 seconds, are given in Figure 5b. It is not surprising that End-system 5 gets a better behavior as it is only connected to one switch. In the next scenarios we shall see that these results can be validated with higher precision and confidence by using SPRT and SSP.

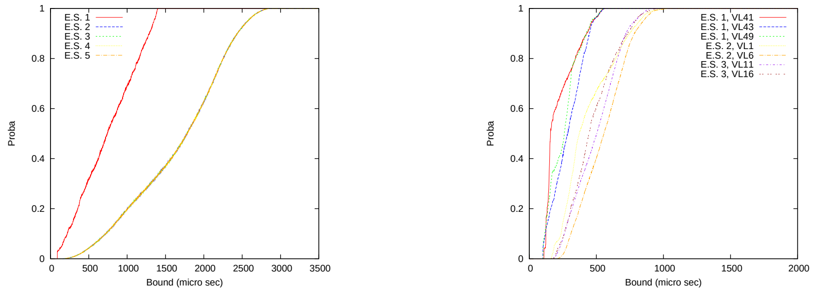
Scenario 2 In this scenario, we consider an increasing number of virtual links: 10, 20, and 30 links. All the virtual links have the same characteristics: BAG = 4ms, and frame size varying between 100 and 500 octets (which is different from the previous experiment in where this number was fixed). Our first experiment was the same as for Scenario 1, except that we had to consider bigger values of the bound in order to reach probability 1. The results are given in Figures 6a, 6b and 7a for respectively $X = 10$, $X = 20$ and $X = 30$ links. We shall observe that the bound varies between $0\mu s$ and $2000\mu s$ for $X = 10$, between $0\mu s$ and $3000\mu s$ for $X = 20$ and between $0\mu s$ and $3500\mu s$ for $X = 30$.



(a) Probability of having a delay lower than a given bound as a function of the bound for $X = 10$ for Scenario 2.

(b) Probability of having a delay lower than a given bound as a function of the bound for $X = 20$ for Scenario 2.

Fig. 6: Illustration of results for Scenario 2.



(a) Probability of having a delay lower than a given bound for $X = 30$ for Scenario 2.

(b) Probability of having a delay lower than a given bound for Scenario 3.

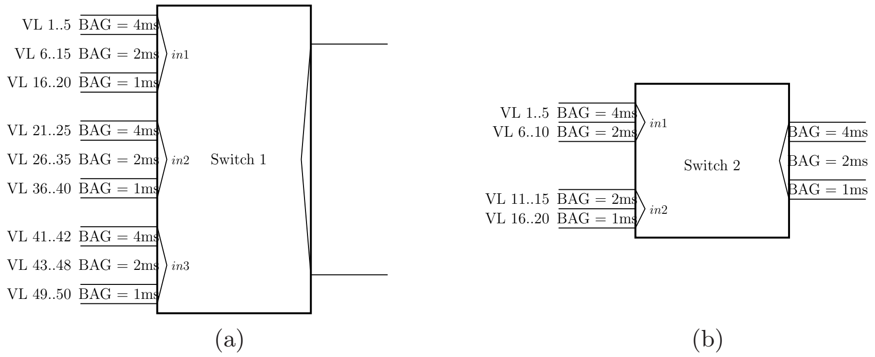
Fig. 7: Illustration of results for Scenario 2 and 3.

The second and third experiment consisted in validating the results we obtained in the first experiment using SPRT and SSP. Those algorithms cannot be used to check for an exact probability ($= \theta$) but for a bound on the probability ($\geq \theta$ or $\leq \theta$ with a simple modification of the algorithm [21]). This is not a problem. Indeed, if ESTIMATION told us that the probability is x for a given bound, then it means then it is also greater or equal to x. As outlined in Section 2.2, working with SSP and SPRT allows us to validate the results with a higher confidence. More precisely, we worked with a precision of 10^{-7} and confidence of 10^{-10} instead of 10^{-2}. The results are given in Table 2 for $X = 10$.

Finally, we have measured the minimum and maximum delay and the jitter (difference between these delays) for one of the virtual link of each End-system. This is a proportion computed on a fixed number of simulations, here $2.3 * 10^9$. We obtained, a jitter of 915 with a maximum delay of 1032 and a minimum delay of 117 for a virtual link between End-system 1 and End-system 6 with $X = 20$.

End-system	Bound	Estimated proba	Algorithm	Checked against	Result	NSimulations
1	479	P = 0.831975	SPRT	P $\geq$ 0.8 ?	Y	1.210^{10}
			SPRT	P $\geq$ 0.85 ?	N	1.510^{-10}
	564	P = 1	SSP	P $\geq$ 1 ?	Y	2.310^{9}
2	859	P = 0.730221	SPRT	P $\geq$ 0.7 ?	Y	1.610^{10}
			SPRT	P $\geq$ 0.5 ?	Y	2.510^{9}
	1064	P = 1	SSP	P $\geq$ 1 ?	Y	2.310^{9}
3	760	P = 0.532258	SPRT	P $\geq$ 0.5 ?	Y	1.910^{10}
			SPRT	P $\geq$ 0.3 ?	Y	2.110^{9}
	1064	P = 1	SSP	P $\geq$ 1 ?	Y	2.310^{9}
4	977	P = 0.931101	SPRT	P $\geq$ 0.9 ?	Y	6.810^{9}
			SPRT	P $\geq$ 0.5 ?	Y	1.310^{9}
	1064	P = 1	SSP	P $\geq$ 1 ?	Y	2.310^{9}
1	512	P = 0.231049	SPRT	P $\geq$ 0.2 ?	Y	1.210^{10}
			SPRT	P $\geq$ 0.1 ?	Y	1.610^{9}
	1064	P = 1	SSP	P $\geq$ 1 ?	Y	2.310^{9}

Table 2: Results of SPRT and SSP experiments for $X = 10$ for Scenario 2.



(a)

(b)

Fig. 8: Repartitions of the virtual links in the switches for Scenario 3.

As the number of simulations is quite high, we believe that our results are quite accurate. This show the interest of working with an executable model.

Scenario 3 In this scenario, we use a fixed number of ten virtual links. Those links have different BAGS, and the packets size still varies between 100 and 500 octets. Figures 8a and 8b represents the repartition of the virtual links in Switches 1 and 2 respectively. Switch 3 has the same repartition of virtual links as Switch 2. We conducted the same experiments as for Scenario 2.

We first measure the probability of having a delay lower than a given bound, varying between $0\mu s$ and $2000\mu s$. The results are given in Figure 7b. We then applied SPRT and SSP. The results are reported in Table 3. All the results are obtained in less than 10 seconds. Finally, we have measured the minimum delay, the maximum delay and the jitter for one virtual link of each group of virtual links with the same characteristics. As an example, we obtain, a jitter of 451 with a maximum delay of 556 and a minimum delay of 105 for VL 49 between End-system 1 and End-system 6.

End-system-VL	Bound	Estimated proba	Algorithm	Checked against	Result	NSimulations
1-41	345	0.801593	SPRT	$P \geq 0.8$	Y	$1.3 \cdot 10^9$
	566	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
1-43	403	0.804363	SPRT	$P \geq 0.8$	Y	$9.2 \cdot 10^8$
	561	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
1-49	347	0.805792	SPRT	$P \geq 0.8$	Y	$9.2 \cdot 10^8$
	556	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
2-1	751	0.901582	SPRT	$P \geq 0.9$	Y	$1.1 \cdot 10^9$
	1044	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
2-6	785	0.902106	SPRT	$P \geq 0.9$	Y	$1.1 \cdot 10^9$
	1051	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
3-11	505	0.504159	SPRT	$P \geq 0.5$	Y	$1.7 \cdot 10^9$
	932	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$
3-16	446	0.502088	SPRT	$P \geq 0.5$	Y	$3.1 \cdot 10^9$
	994	1	SSP	$P \geq 1$	Y	$2.3 \cdot 10^9$

Table 3: Results of SPRT and SSP experiments Scenario 3.

6 Conclusion and Related Work

This paper proposes a model of the AFDX network based on the compositional design approach as well as a verification technique based on statistical model checking. To the best of our knowledge, this is the first complete, fully operational and timing accurate, model of AFDX developed using a formal framework. Other models are either performance models built within network simulators or timed automata models, restricted to few functionalities or describing very simple network configuration. The work of [4] focused on redundancy management and identified several issues occuring in the presence of particular network faults. Alternatively, [7, 8, 18] deal with computing bounds for end-to-end delays in AFDX networks. The papers [7, 8] report experiments using three analysis methods: network calculus, stochastic simulation using QNAP2 and timed model-checking using Uppaal. The results confirm the well-established knowledge about these methods. Network calculus[10] provides pessimistic, unreachable bounds. Network stochastic simulation provide reachable bounds, however, these bounds hardly depend on the simulation scenario considered and can be too optimistic. Timed model-checking[3] provide exact bounds, however, it suffers for state explosion due to model complexity, and hence, cannot scale to realistic networks. Finally, the work in [18] provides a method for compositional analysis of end-to-end delays. It is shown that, to measure delays for a given virtual link, it is enough to consider only the traffic generated by the virtual links influencing, i.e., which share paths within the network. This observation allows to *slice* the network and therefore to reduce the complexity of any forthcoming analysis. However (1) our model is more detaillel and easier to extend/modify due to the use of the component-based design approach, (2) we are capable to retreive stochastic informations on the network, (3) our approach is adaptive.

References

1. ARINC 429, Aeronautical Radio Inc. ARINC specification 429. Digital Information Transfer Systems (DITS) part 1,2,3. (2001)

2. ARINC 664, Aircraft Data Network, Part 7: Avionics Full Duplex Switched Ethernet (AFDX) Network. (2005)
3. Alur, R., Dill, D.: A Theory of Timed Automata. Theoretical Computer Science 126, 183–235 (1994)
4. Anand, M., Dajani-Brown, S., Vestal, S., Lee, I.: Formal modeling and analysis of afdx frame management design. In: ISORC. pp. 393–399. IEEE (2006)
5. Basu, A., Bozga, M., Sifakis, J.: Modeling Heterogeneous Real-time Systems in BIP. In: SEFM06, Pune, India. pp. 3–12 (September 2006)
6. Basu, A., Bensalem, S., Bozga, M., Caillaud, B., Delahaye, B., Legay, A.: Statistical abstraction and model-checking of large heterogeneous systems. In: FORTE 2010. pp. 32–48. LNCS 6117, Springer-Verlag (2010)
7. Charara, H., Fraboul, C.: Modelling and simulation of an avionics full duplex switched ethernet. In: Proceedings of the Advanced Industrial Conference on Telecommunications/ Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunication Workshop. IEEE (2005)
8. Charara, H., Scharbarg, J.L., Ermont, J., Fraboul, C.: Methods for bounding end-to-end delays on AFDX network. In: ECRTS. IEEE Computer Society (2006)
9. Clarke, E.M., Faeder, J.R., Langmead, C.J., Harris, L.A., Jha, S.K., Legay, A.: Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In: CMSB. LNCS, vol. 5307, pp. 231–250. Springer (2008)
10. Cruz, R.: A calculus for network delay. IEEE Transactions on Information Theory 37(1), 114–141 (January 1991)
11. Grosu, R., Smolka, S.A.: Monte carlo model checking. In: TACAS. LNCS, vol. 3440, pp. 271–286. Springer (2005)
12. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: VMCAI. LNCS, vol. 2937, pp. 73–84. Springer (2004)
13. Hoeffding, W.: Probability inequalities. Journal of the American Statistical Association 58, 13–30 (1963)
14. Jansen, D.N., Katoen, J.P., M.Oldenkamp, Stoelinga, M., Zapreev, I.S.: How fast and fat is your probabilistic model checker? an experimental performance comparison. In: HVC. LNCS, vol. 4899. Springer (2007)
15. Jha, S.K., Clarke, E.M., Langmead, C.J., Legay, A., Platzer, A., Zuliani, P.: A bayesian approach to model checking biological systems. In: CMSB. LNCS, vol. 5688, pp. 218–234. Springer (2009)
16. Laplante, S., Lassaigne, R., Magniez, F., Peyronnet, S., de Rougemont, M.: Probabilistic abstraction for model checking: An approach based on property testing. ACM Trans. Comput. Log. 8(4) (2007)
17. Legay, A., Delahaye, B.: Statistical model checking : An overview. CoRR abs/1005.1327 (2010)
18. Scharbarg, J.L., Fraboul, C.: Simulation for end-to-end delays distribution on a switched ethernet. In: ETFA. IEEE (2007)
19. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: CAV. pp. 202–215. LNCS 3114, Springer (2004)
20. Wald, A.: Sequential tests of statistical hypotheses. Annals of Mathematical Statistics 16(2), 117–186 (1945)
21. Younes, H.L.S.: Verification and Planning for Stochastic Processes with Asynchronous Events. Ph.D. thesis, Carnegie Mellon (2005)
22. Younes, H.L.S., Kwiatkowska, M.Z., Norman, G., Parker, D.: Numerical vs. statistical probabilistic model checking. STTT 8(3), 216–228 (2006)