

Towards Reliability Evaluation of AFDX Avionic Communication Systems With Rare-Event Simulation

Armin Zimmermann^{a*}, Sven Jäger^a, and Fabien Geyer^b

^aSoftware and Systems Engineering, Ilmenau University of Technology; Ilmenau, Germany

^bAirbus Group Innovations, Dept. TX4CP; Munich, Germany

Reliability is a major concern for avionic systems. The risks in their design can be minimized by using model-based systems engineering methods including simulation and mathematical analysis. However, there are non-functional properties that are computationally expensive to evaluate, for instance when rare events are important. Rare-event simulation methods such as RESTART can be used, leading to speedups of several orders of magnitude. We consider AFDX (avionic full-duplex switched Ethernet) networks as an application example here, where the end-to-end delay and buffer utilizations are important for a safe and efficient system design. The paper proposes generic model patterns for AFDX networks, and shows how very low probabilities can be computed in acceptable time with the presented method and software tool.

Keywords: Rare-Event Simulation, AFDX, Avionic Networks, Stochastic Petri Nets

1. INTRODUCTION

Reliability and safety are important non-functional requirements of many man-made systems, especially when failures may lead to catastrophic events. Common examples include automotive systems, train control, and avionics. The resulting effect of local design decisions on overall system properties are not obvious, because there are numerous specialists working on design details. Mathematical models can help to describe such systems and to compute their system properties with the help of appropriate software tools (“model-based design” and “model-based systems engineering” [1]).

Unavoidable faults may be masked or tolerated by static or dynamic redundancy measures. The main task is to design a system such that its reliability and safety requirements are achieved with the least amount of resources. Classic models and tools for static analysis such as Fault Trees and Reliability Block Diagrams [2] are wide-spread in these domains, but are not able to cover systems in which the complex behavior influences failures, or if dynamic reconfigurations are applied. In avionic system design, fly-by-wire systems, flight control and management, maintenance processes, as well as future communication architectures are examples in which dynamic reliability models are necessary.

Avionic networks are an increasingly important element of distributed sensing, processing, and control architectures on board modern aircrafts [3]. A modern avionic communication system is AFDX ([4], more details in Section 2). Its general reliability aspects based on hardware failures is important for system design and certification, and can be analysed with models [5]. It uses a dual-layer hardware redundancy setup (static redundancy) to survive single failures of hardware elements.

In this paper we are interested in the more complex question of network guarantees for the served real-time applications, which require a model-based analysis of end-to-end delays [6, 7, 8]. There are several methods for worst-case end-to-end delay analysis (simulation, network calculus, and model checking [6]), all with their individual advantages and drawbacks. Besides the concentration on guaranteed bounds on maximum end-to-end delays, an upcoming question for network and buffer

* Corresponding author, armin.zimmermann@tu-ilmenau.de

sizing are probabilistic end-to-end measures such as quantiles of the distribution [9]. For instance, the pessimism of bounds may lead to system designs in which the guaranteed maximum delay may be 20ms, while the actually observed delay rarely exceeds 1ms.

The end-to-end delay evaluation can be reduced to an analysis of buffer levels and their probabilities (c.f. Section 2), which this paper concentrates on. Buffer overflow with packets is guaranteed to not happen in switches designed based on guarantees, but it is interesting to check buffer utilization and what the actual probabilities of utilized buffers are. Switch design can benefit from knowing how many buffer elements are needed. Moreover, if very rare packet losses or delays exceeding the guarantee are acceptable by the served applications, how much buffer space can be saved? There is obviously a trade-off between resource utilization (and cost) vs. end-to-end delays (c.f. Sections 2 and 4).

The necessary dynamic models need to consider discrete events, states, probabilistic choice and stochastic delays. Depending on the complexity of the system behavior and the corresponding size of the state space, simulation programs, Markov chains, and stochastic Petri nets (SPNs) are applied to reliability problems in the literature [2], among others. The latter two are attractive as long as the underlying assumption of a Markov behavior is realistic, because then a direct numerical solution is possible [10]. Petri nets have been suggested for reliability engineering of complex systems in an international standard recently [11].

However, non-Markovian delay distributions are necessary, for instance, in the case of periodic events typical of AFDX networks and embedded systems in general. The numerical analysis of models incorporating them is very restricted, only allowing the application to special cases [10]. An alternative evaluation technique is simulation, but the problem here is that the computational effort to generate enough failure states to achieve statistical confidence in the estimated results is usually intractable — it simply takes too long until significant events are generated.

This problem is well-known as rare-event simulation, and there are two main approaches used: *importance sampling* and *splitting*. They have the common goal to increase the frequency of the rare event in order to gain more significant samples out of the same number of generated events. Among methods that can be automated and implemented in a software tool for industrial applications, the splitting technique has the advantage of requiring less insight into the model details. A variant is the RESTART algorithm [12], which has been shown to work robustly and efficiently for many applications. Considerable speedups of several orders of magnitude can be achieved even for non-trivial system models. Rare-event simulation of general communication networks with importance sampling is, for instance, presented in [13].

A brief description of AFDX networks and related work on its model-based design and analysis is given in the subsequent section. After a short coverage of stochastic Petri nets in Section 3, generic patterns for AFDX network modeling with SPNs are proposed in Section 3.1. The topology of an AFDX application example network is presented in Section 3.2 together with its SPN model, which has been constructed modularly with the patterns. Section 4 explains how the used rare-event simulation technique RESTART works, points out the used software tool TimeNET [14], and presents numerical results of simulation experiments carried out for the example with it.

The contribution of the paper are realistic SPN model patterns for AFDX, and to show that existing methods in rare-event simulation can help to compute reliability measures that are otherwise computationally intractable. To the best of the authors' knowledge, this technique has not been applied in avionics reliability evaluation before. Results are presented for an example with non-trivial size.

AFDX network modeling and performance evaluation using stochastic Petri nets has been tried before [15, 16]. However, no network structure has been taken into consideration; the transmission delay is assumed to be a sequence of exponential transitions only, independent of the actual number of links. The load model is assumed as a mix of periodic and sporadic message generations that alternate. The mean end-to-end delay is analyzed based on this overly simplified model in [15]; however, more detailed information about its distribution such as quantiles or maximum values are of much higher interest.

2. MODEL-BASED DESIGN OF AFDX NETWORKS

AFDX (Avionics Full-Duplex Ethernet) is a data network based on Ethernet, developed by Airbus and created during the development of the A380. It was standardized in Part 7 of the ARINC 664 specifications [4], and has since then been used in other Airbus projects.

This network technology is based on switched Ethernet twisted pair 100 Mbps full duplex technology. It attempts at addressing the issue of non-deterministic network, best-effort and lack of bandwidth guarantees of traditional Ethernet. It aims at providing a redundant deterministic network, adapted to safety-critical applications used in aircrafts. The main differences compared to Ethernet, are the redundancy property where frames are duplicated and sent on two separate networks, a frame identifier at layer 2 to avoid packet duplication, and a verification of flow properties (packet size and frequency) by the switches. The nondeterministic effect of message collisions on standard Ethernet is avoided by connecting only two nodes with each physical link and using a dedicated link for each direction (full duplex). Thus there will be no collisions on the physical level, and the only nondeterminism can arise from the possible waiting times in output queues of switches because of temporary link contention.

An AFDX network is composed of *end-systems* and *switches* as nodes. End-systems serve as source and destination nodes in the network, over which applications may send data according to bandwidth restrictions to avoid overloading. One fundamental building block of AFDX is the notion of *virtual link* (VL), which can be seen as rate-constrained network tunnels. The parameters describing a VL are: the emitter end-system of this VL, the list of receiving end-systems, static routes between emitter and receivers, the Bandwidth Allocation Gap (BAG), as well as minimum and maximum frame length (s_{min} and s_{max}). The BAG is defined as the minimum time interval between the first bit of two consecutive frames from the same VL and has a value of $2^k ms$ with $k \in \{1..7\}$.

The packet structure follows Ethernet and contains 67 Bytes overhead (including the inter-frame gap) in addition to the possible 17...1471 Bytes payload (between which s_{min} and s_{max} can be chosen). Assuming a transmission bandwidth of 100 Mbps, each packet will thus require a per-link transmission time between $6,72\mu s$ and $123.04\mu s$.

The elements of the AFDX network are deterministic, the only source of randomness is in the times that end-systems have to send packets (or the offsets between them). Even if every application would be sending periodic messages only with the maximum frequency given by its BAG value, there is no globally synchronized clock and thus any offset between end-systems may occur already because of clock drift. Sporadic message generations can happen at arbitrary times, as they can be sent immediately after generation if the last message has been sent more than the BAG value before.

Important properties of an avionic network are safety against packet loss (by avoiding buffer overruns and redundant hardware) as well as a maximum end-to-end delay (specified dependent on the network architecture [4]). The guaranteed worst-case behavior of AFDX comes from the encapsulation of every network flow in a VL, and the fact that the VL properties are enforced by the switches in the network.

If an end-system does not send packets according to the VL specifications (BAG and frame size), the packets are dropped, which avoids overloading the network and guarantees the end-to-end latencies of the other flows.

Elements of the end-to-end delay that a packet experiences are discussed in [17]. There are unavoidable deterministic parts: 1) the transmission delay over the statically predefined set of links for a VL and 2) processing delays in switches between their input and output ports (hardware- and implementation-dependent, but guaranteed not to exceed $16\mu s$). The sum of these delays constitutes a minimum transmission delay in the case of no queuing.

However, temporarily the network may be populated because of resource sharing: if a packet is put into a switches' output buffer and finds the subsequent transmission link busy, or even other packets in front of it in the queue, there will be a delay before the packet may be transmitted. These delays lead to jitter in the overall end-to-end delay, and are thus the subject of several analysis approaches in the literature. The most important property for a certification of an AFDX network for flight-critical applications is a guaranteed maximum end-to-end delay.

The most prominent methods are standard and stochastic network calculus, which allow to compute safe upper bounds on the maximum end-to-end delay for industrial-size network topologies [7, 18, 6]. The algorithm can be improved with the trajectory approach [8]. For small-size systems the state space may be manageable, allowing to compute an actual maximum delay with model checking [19]. Simulation is another choice [20, 17], but there is no guarantee that the visited parts of the stochastic process will include the worst-case delay. It will, however, give a lower bound on possible maximum delays.

The quality of computable bounds is discussed in the literature [7]: simplified, the derived bounds are less tight (the pessimism increases) when the network topology becomes larger, and with higher network loads [17]. They are best if only one switch is used, but may be off by a factor of up to 20 otherwise. However, industrial-size networks contain paths with up to 4 switches [17]. Unfortunately, the worst case cannot be derived by simply assuming worst-case input values; the end-to-end delay increases for some cases, when the BAG occupation of another VL is decreased [7].

The downside cost of a provably safe network setup with some remaining pessimism that is never needed in reality leads to a bad utilization of network resources. The maximum utilization of real-life AFDX networks is usually around or below 20%. Another issue is that even if a computed bound is tight, the probability that a packet will actually experience it may be marginally small and acceptable for the applications waiting for it. There is thus an interest in not only computing bounds on the worst case, but also the actual end-to-end delay distribution or its quantiles, as well as the connected probabilities of a certain buffer utilization [21]. It is, however, still an open problem how resources can be better utilized depending on how rare the computed maximum delays are [9]. A possible solution approach is presented in this paper in Section 4.

3. AFDX NETWORK MODELING WITH STOCHASTIC PETRI NETS

Stochastic Petri nets (see [22, 23], e.g., for an overview) represent a graphical and mathematical method for the specification of processes with concurrent, synchronized and conflicting or nondeterministic activities. The graphical representation of Petri nets comprises only a few basic elements. They are therefore useful for documentation and a figurative aid for communication between system designers. Complex systems can be described in a modular way, where only local states and state changes need to be considered. The mathematical foundation of Petri nets allows their qualitative analysis based on state equations or reachability graph, and their quantitative evaluation based on the reachability graph or by simulation.

Petri nets contain *places* (depicted by circles), *transitions* (depicted by boxes or bars) and directed *arcs* connecting them. Places may hold *tokens*, and a certain assignment of tokens to the places of a model corresponds to its model state (called *marking* in Petri net terms). Transitions model activities (state changes, events). Just like in other discrete event system descriptions, events may be possible in a state — the transition is said to be *enabled* in the marking. If so, they may happen atomically (the transition *fires*) and change the system state.

In stochastic Petri nets, activities may take some time, thus allowing the description and evaluation of performance-related issues. Basic quantitative measures like the throughput, loss probabilities, utilization and others can be computed. A *firing delay* is associated to each transition, which may be stochastic (a random variable) and thus described by a probability distribution. It is interpreted as the time that needs to pass between the enabling and subsequent firing of a transition. In the net class *extended deterministic and stochastic Petri nets* (eDSPN [10]) that is used here, transition delays may be zero (immediate), exponentially distributed, deterministic, or a general distribution can be specified.

The dynamics of a Petri net are defined as follows. A transition is said to be *enabled* in a marking, if there are enough tokens available in each of its input places. Whenever a transition becomes newly enabled, a *remaining firing time* (RFT) is randomly drawn from its associated firing time distribution. The RFTs of all enabled transitions decrease with identical speed until one of them reaches zero (race enabling semantics). The fastest transition (in case of multiple ones, a probabilistic choice decides) will *fire*, and change the current marking to a new one by removing the necessary number of tokens from the input places and adding tokens to output places.

3.1. Petri Net Patterns for AFDX Network modeling

One of the advantages of Petri net models over, e.g., automata, is their modular way of describing complex systems. The following text proposes generic Petri net patterns for AFDX elements along their functional details as described in Section 2.

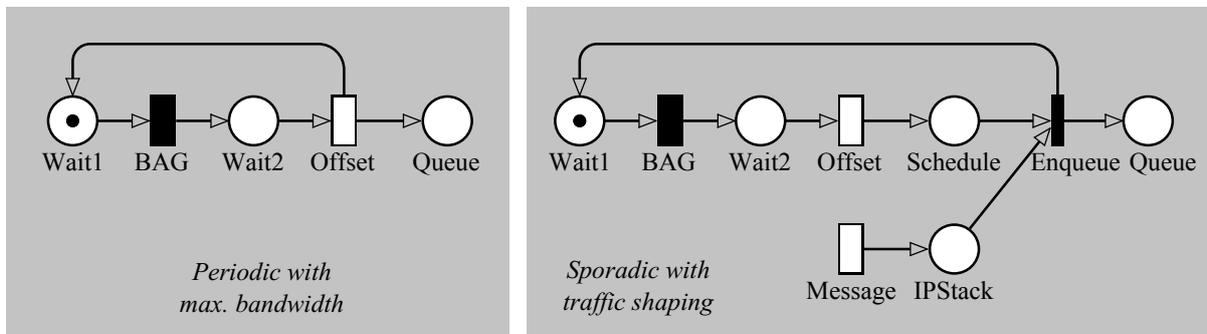


Figure 1: Petri net model variants of AFDX source end-system with packet regulation

Figure 1 introduces two variants of source end-systems. The left part shows how applications and virtual links generating periodic messages should be modeled. Every time both transitions `BAG` and `Offset` have fired sequentially, one packet will be generated and the according token is added to place `Queue`. FIFO behavior of queuing is not significant here, as there is no way of (and no need to) differ between tokens resp. messages under certain simplifying assumptions.

It may seem strange at first that a periodic behavior is modeled by a deterministic plus an exponential transition. This is however necessary to create a stochastic process covering all possible end-system offsets in a steady-state simulation run. The exponential transition `Offset` models drift and other

possible inaccuracies between end-system clocks in the overall distributed system, because there is no global time synchronization [24].

The right part of Figure 1 shows a model variant in which an application sends sporadic messages over a virtual link (transition `Message` adds a token to `IPStack`), and thus the end-system needs to apply traffic shaping to ensure a minimum BAG time between messages. This is done similarly to the left-hand side model. When a message may be added to the output queue of the end-system, a token is in `Schedule`, and transition `Enqueue` will fire immediately when a message arrives in `IPStack`. After that, transitions `BAG` and `Offset` have to fire subsequently before a new message may pass.

In the case of a VL carrying periodic messages, but with a longer time between subsequent message generations than the BAG, the left-hand model in Figure 1 can still be used, but the firing delay of transition `BAG` would have to be set to this fixed time between messages. In any case, as long as the period does not exceed the allowed BAG value, traffic shaping will not change the behavior and is thus not explicitly modeled.

In general, transition delays in the models have to be set according to the chosen model time as well as the delays of the described actions. We assume a base time of 1 model time unit equal to $1\mu s$ in the following. Possible BAG values in the range $1ms \dots 128ms$ will then result in transition delays $1,000 \dots 128,000$. The actual value of the random offset is not that important, but should be small compared to the BAG¹. The average time between sporadic message generations in this case is associated with transition `Message`.

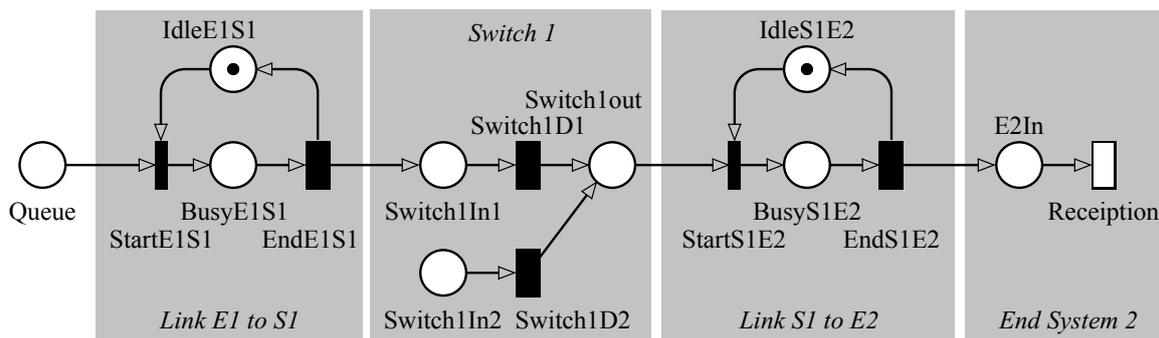


Figure 2: Link, switch, and destination end system model

Petri net patterns for links between nodes, switches, and destination end-systems are proposed in Figure 2. Larger models can be constructed by merging corresponding places such as `Queue` from both model figures. A link is simply a mutually exclusive resource for queued waiting messages. It is either in state `Idlexy` or `Busyxy`. A transmission begins immediately (`Startxy` fires), when a message is waiting and the link is idle. The transmission time for each message is modeled by the delay of the deterministic transition `Endxy`, which is selected based on message length and bandwidth. For the minimum and maximum message lengths (84 and 1538 Bytes), the delay has thus to be chosen as 6.72 and 123.04 in our μs -based timing. We assume identical message lengths here.

The switch has input and output ports (places `SwitchInx` and `SwitchOutx` in Figure 2, for instance). Any number of input ports for incoming links can be added similar to the two shown in

¹ For an even better approximation, the mean delay could be chosen depending on the variance in the end-system clocks, and its half subtracted from the BAG to compute the firing delay of transition `BAG`.

the model. The $16\mu s$ delay inside the switch to process a message and enter it into an output port is modeled by transitions `SwitchDx`. A second link connects the switch in the model with a destination end-system.

3.2. An Application Network Setup

Figure 3 sketches a sample topology of an AFDX network that has been chosen as an application example. Its design resembles network architecture examples in the literature [8] with a little longer switch sequence but without VL paths that split away from the considered flows at switches².

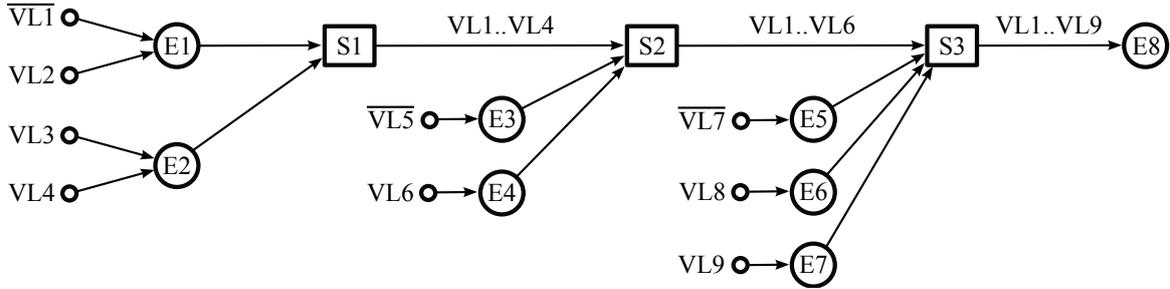


Figure 3: Topology of a sample AFDX network

A real-life topology would not necessarily contain longer paths (a real-life size avionic network with several thousand VLs analyzed in [17] had only four switches on the longest paths), but many more VLs and paths. However, following the findings of [17], only VLs on paths that *directly influence* the VL or path under consideration have to be taken into account, as the others do not influence the end-to-end delay distribution. This allows to ignore all VLs which do not share an output port with the analyzed VL at any switch in the network, and decreases the size of the significant network substantially.

The example topology contains 8 end-systems $E1..E8$, 3 switches $S1..S3$, and 10 links between them. 9 virtual links $VL1..VL9$ are carried by the network, their association to end-systems and links is obvious from Figure 3. The overlined virtual links $\overline{VL1}$, $\overline{VL5}$ and $\overline{VL7}$ symbolize periodically sending applications; i.e., a packet is sent with every BAG. The remaining VLs carry sporadic traffic, which is assumed to be requested by the application randomly (at a lower rate than $1/BAG$ to avoid overloading), but needs to be controlled and possibly delayed before entering the network.

Figure 4 depicts the full eDSPN model constructed with the tool TimeNET [14]. It is designed in a modular way using the building blocks proposed in Section 3. BAG values are set to the minimal value of $1ms$ equal to a transition delay of 1000 (with an additional jitter of $5\mu s$ at transitions `Offset`). Messages on sporadic virtual links are sent on average every $1.2ms$. Packet lengths are chosen as 8000bits, corresponding to a transmission delay of $80\mu s$. The delay spent by each message in a switch between arrival and queuing or transmission is 16 (transitions `Switchx`).

4. RARE-EVENT SIMULATION FOR AFDX BUFFER SIZING

We are interested in probabilities that the number of packets in a network buffer exceeds certain values in stationary operation, and these states of interest only happen rarely in comparison to the rapid state changes inside the network. Thus a regular simulation would have to execute huge numbers

² The reason for this choice is that for such a system setup, packets arriving from one input of a switch but belonging to different VLs would have to be queued at different output queues. This would require additional information which is not available with the standard Petri nets chosen here.

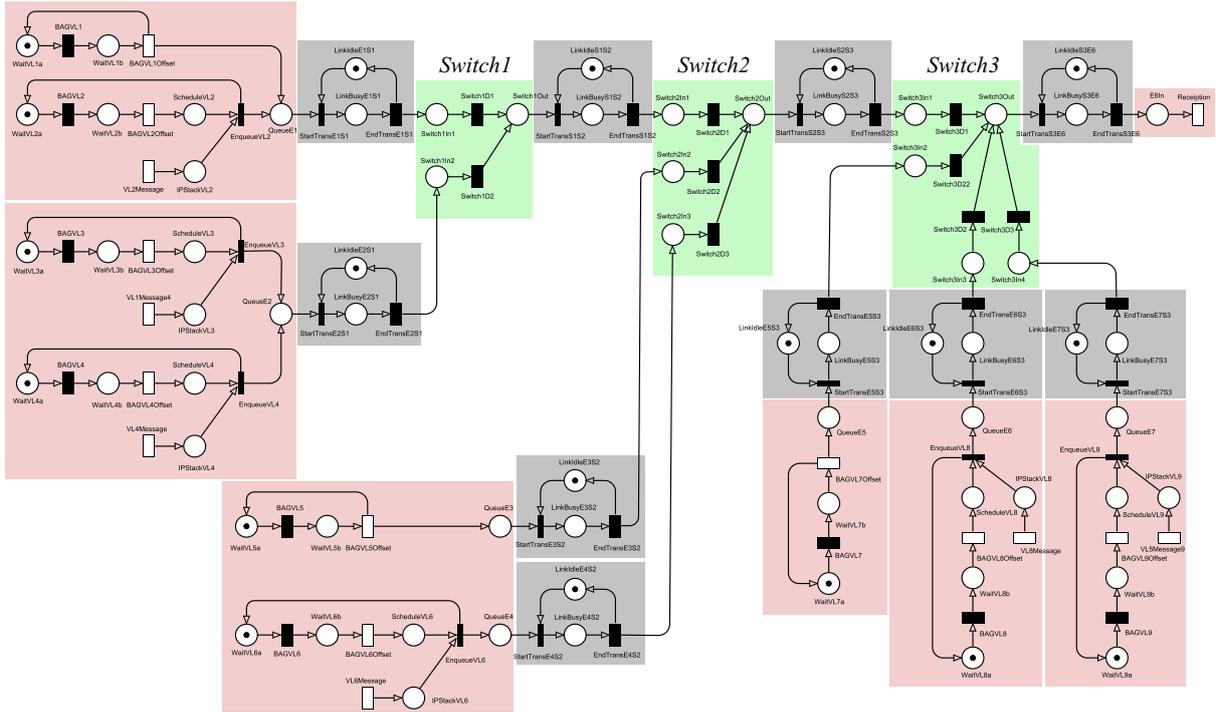


Figure 4: A Petri net model of the example avionic communication system

of events until a sufficiently large number of significant events has been collected and the estimated performability values have converged with a predefined error margin.

The section briefly touches the RESTART method used in this paper to solve this issue, and shows how it can be applied to our application model.

4.1. The RESTART Algorithm

The RESTART algorithm [12] cuts the reachability space of a model into enclosing regions of increasing probability to hit a rare event or state. Upon entering a state in a region of higher probability (“closer” to the region of interest), the state is stored and may be restored later when the simulation state is about to leave the region. Applying this simple rule on every border between regions leads to a simulation that hits the rare events of interest much more often. Some changes have to be applied to the accumulation of performance measures during the simulation to reverse the bias introduced by this way of controlling the simulation trajectory. A *weight* is maintained by each simulation path, which starts with 1.0 and is divided by the *splitting factor* whenever the path crosses a region border upwards. The factor equals the number of times that the path is retried from a stored state, which leads to a setup similar to the splitting of particles. The final surviving path after a split will be continued, and regains the previous weight.

A user-specified *importance function* returns a number for each state, which should be related to the distance of the current state from the region of interest. In our application model, simply the number of tokens in the buffer `Switch3Out` is used, which is actually not a very good measure for our RESTART application, but it turns out that the speedup is still very high - underlining the robustness of the approach. RESTART for stochastic Petri nets has been proposed in [25] and improved later [23, 26].

The issue of rareness in simulation is comparable to the problem that has been identified in the literature for improvements of network calculus that considering trajectories with expected long delays [6, 27]. An interesting question for future research is how the existing knowledge about trajectories leading to near-worst-case behavior could be used for the definition of a better importance function for RESTART.

We use the tool TimeNET [14, 23] here, which implements several model classes of stochastic Petri nets and their analysis and simulation, as well as RESTART splitting for reliability measures [28, 26].

4.2. Evaluation of the Application Example

This section presents some performability results for the AFDX network model shown in Figure 4. All simulations have been carried out on an Intel core i5 2.4GHz laptop computer running Windows 7 64bit. Simulation accuracy has been chosen as confidence level 95%; the maximum allowed relative error is 10%, and detection of the initial transient is activated in the TimeNET simulation algorithms.

All traffic ends at end-system E_8 in our model, and thus the most heavily loaded link is the one from switch S_3 to it. Regular simulation computes the link utilization to be 0.6507, and the mean number of messages waiting in the queue (tokens in place `Switch3Out`) as 0.2963 (i.e., waiting in addition to a currently transmitted message). To validate the results, the utilization can be compared to a theoretical approximation: there are 3 VLs sending (almost) periodically, occupying the link for $80\mu s$ within a time period of $1005\mu s$. In addition, there are 6 sporadically sending VLs, transmitting an $80\mu s$ -packet every $1200\mu s$ on average. This corresponds to a theoretical joint utilization of 0.6376, resulting in an error of about 2% compared to the simulation result.

For our buffer sizing problem and an analysis of the variable waiting time of messages because of queuing, we are interested in the probabilities of having at least n messages waiting in place `Switch3Out`. The results of this analysis are shown in Table 1.

Number of Messages	Standard Simulation		RESTART	
	Probability	CPU time	Probability	CPU time
1	2.6221E-01	0:00:01	2.30463E-01	0:00:03
2	4.3991E-02	0:00:05	3.00958E-02	0:00:15
3	2.1414E-03	0:04:56	2.32636E-03	0:04:39
4	—	>24h	8.54219E-21	1:04:37

Table 1: Probabilities of buffered messages exceeding bounds (time in h:min:sec)

The table shows the net required CPU time computed by normal and RESTART simulation. As the tool uses a master/slave process architecture with 3 slaves to decrease variance with multiple independent replications, the multiple cores of the computer can be utilized, and thus the actual run time of each simulation experiment is only about a third of the shown value.

The results show that up to three tokens / messages, the probabilities are getting smaller but stay in a range that is not rare. Therefore, both normal and RESTART simulation are able to compute the values with comparable and acceptable CPU times. The reason for this is that this value is related to the probability of having an arriving packet on a subset of the four input links to switch 3 during overlapping time intervals. However, for token numbers above three, the probability drops considerably,

and cannot be computed with normal simulation in an acceptable time³. The probability that the number of waiting messages at switch 3 is bigger than 3 is less than 10^{-20} , and this very rare value can be derived by the RESTART simulation method after about one hour with the same convergence requirements as for the other experiments. This shows that even for complex system models which are unfavorable for the RESTART algorithm, considerable speedups can be achieved.

The results show that while for certification purposes of safety-critical applications a mathematical proof of worst-case assumptions may be legally necessary, it is possible to evaluate the actual probabilities of exceeding certain buffer utilization values and to decide how many buffers are actually needed. In our example, if the output buffer of switch 3 would be restricted to 3 slots, the resulting loss probability of incoming packets would be estimated in the order of magnitude of 10^{-18} , corresponding to one lost message within about every 4 million years operation time. This will most probably be acceptable because of the other layers of redundancy in the used avionic hard- and software. Moreover, the value is negligible small compared to the packet error rate of Ethernet, which is around $10^{-9} \dots 10^{-12}$.

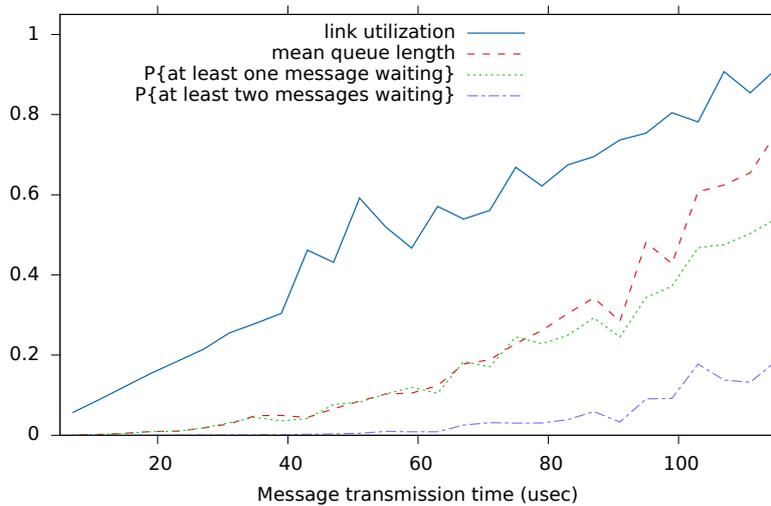


Figure 5: Various performance measures of link S3E6 vs. message length

A second experiment was conducted to find out the dependency of basic network performance measures on the overall utilization and validate model as well as simulation. Link utilization can be changed without adding more VL structural models by simply varying the message length of the existing model within the allowed range. The results are shown in Figure 5. The expected result of a linear dependency of link utilization is visible (within the bounds of simulation inaccuracy). Queue lengths and probabilities of having at least one or two messages waiting at the output port of switch 3 are increasing nonlinearly, as it can be expected from queuing theory.

5. CONCLUSION

The paper shows how AFDX networks can be modeled with stochastic Petri nets and proposes a set of realistic patterns. It demonstrates use and advantages of the splitting rare-event simulation method RESTART by applying it to a non-trivial AFDX network example and deriving probabilities in the range of 10^{-20} in acceptable simulation time.

³ The corresponding experiment has been stopped after more than 8 hours without hitting even one significant event, which is equal to more than 24 hours of CPU time.

Acknowledgements

The authors would like to acknowledge the work of Alexander Wichmann and Timur Ametov, who implemented the current RESTART algorithms in TimeNET. This work has been supported by a TU Ilmenau internal excellency grant in funding period 2013/14.

REFERENCES

- [1] A. Ramos, J. Ferreira, and J. Barcelo, "Model-based systems engineering: An emerging approach for modern systems," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 1, pp. 101–111, January 2012.
- [2] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Wiley, 2002.
- [3] T. Schuster and D. Verma, "Networking concepts comparison for avionics architecture," in *IEEE/AIAA 27th Digital Avionics Systems Conference (DASC 2008)*, 2008, pp. 1–11.
- [4] "Arinc 664, aircraft data network, part 7: Avionics full duplex switched Ethernet (AFDX) network," Jun. 2005.
- [5] K. Wang, S. Wang, and J. Shi, "Integrated reliability theory and evaluation methodology of AFDX," in *10th IEEE Int. Conf. on Industrial Informatics (INDIN)*, 2012, pp. 657–662.
- [6] J.-L. Scharbag and C. Fraboul, "Methods and tools for the temporal analysis of avionic networks," in *New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems*, M. J. Er, Ed. Sciyo, 2010.
- [7] H. Charara, J.-L. Scharbag, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network," in *Proc. 18th Euromicro Conf. on Real-Time Systems (ECRTS06)*, 2006.
- [8] H. Bauer, J.-L. Scharbag, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *IEEE Trans. Industrial Informatics*, vol. 6, no. 4, pp. 521–533, Nov. 2010.
- [9] C. Fraboul and J.-L. Scharbag, "Trends in avionics switched Ethernet networks," in *Proc. 1st Workshop on Real-Time Ethernet (RATE) at the IEEE Real-Time Systems Symposium*, Vancouver, Canada, 2013.
- [10] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.
- [11] "Analysis techniques for dependability — Petri net techniques," IEC 62551:2012, Sep. 2013.
- [12] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of RESTART simulation: Theoretical basis and sensitivity study," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 373–385, 2002.
- [13] J. K. Townsend, Z. Haraszti, J. A. Freebersyser, and M. Devetsikiotis, "Simulation of rare events in communications networks," *IEEE Communications Magazine*, vol. 36, no. 8, pp. 36–41, 1998.
- [14] A. Zimmermann, "Modeling and evaluation of stochastic Petri nets with TimeNET 4.1," in *Proc. 6th Int. Conf on Performance Evaluation Methodologies and Tools (VALUETOOLS)*. IEEE, 2012, pp. 54–63.

- [15] Z. Jiandong, L. Dujuan, and W. Yong, "Modelling and performance analysis of AFDX based on Petri net," in *2nd Int. Conf. on Future Computer and Communication (ICFCC)*, vol. 2, May 2010, pp. 566–570.
- [16] D. Li, J. Zhang, and B. Liu, "Periodic message-based modeling and performance analysis of AFDX," in *IEEE Int. Conf. on Wireless Communications, Networking and Information Security (WCNIS)*, 2010, pp. 162–166.
- [17] J.-L. Scharbag, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an AFDX avionic network," *IEEE Trans. Industrial Informatics*, vol. 5, no. 1, pp. 38–49, February 2009.
- [18] T. Lv, N. Hu, Z. Wu, and N. Huang, "The analysis of end-to-end delays based on AFDX configuration," in *9th Int. Conf. on Reliability, Maintainability and Safety (ICRMS)*, 2011, pp. 1296–1300.
- [19] M. Adnan, J.-L. Scharbag, J. Ermont, and C. Fraboul, "An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows," in *Proc. 16th IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Sep. 2011, pp. 1–4.
- [20] J.-L. Scharbag and C. Fraboul, "Simulation for end-to-end delays distribution on a switched Ethernet," in *Proc. IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2007)*, 2007, pp. 1092–1099.
- [21] H. Bauer, J. Scharbag, and C. Fraboul, "Worst-case backlog evaluation of avionics switched Ethernet networks with the trajectory approach," in *24th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2012, pp. 78–87.
- [22] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Series in parallel computing. John Wiley and Sons, 1995.
- [23] A. Zimmermann, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.
- [24] R. Alena, J. Ossenfort, K. Laws, A. Goforth, and F. Figueroa, "Communications for integrated modular avionics," in *IEEE Aerospace Conference*, March 2007, pp. 1–18.
- [25] C. Kelling, "Rare event simulation with RESTART in a Petri net modeling environment," in *Proc. of the European Simulation Symposium*, Erlangen, 1995, pp. 370–374.
- [26] A. Zimmermann and P. Maciel, "Importance function derivation for RESTART simulations of Petri nets," in *9th Int. Workshop on Rare Event Simulation (RESIM 2012)*, Trondheim, Norway, Jun. 2012, pp. 8–15.
- [27] E. Heidinger, "Rare events in network simulation using MIP," in *Proc. 23rd Int. Teletraffic Congress (ITC 2011)*, 2011, pp. 314–315.
- [28] A. Zimmermann, "Dependability evaluation of complex systems with TimeNET," in *Proc. Int. Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS 2010)*, Valencia, Spain, Apr. 2010.