The Wireless Fire Alarm System: Ensuring Conformance to Industrial Standards through Formal Verification

Sergio Feo-Arenis, Bernd Westphal, Daniel Dietsch,
Marco Muñiz, and Siyar Andisha

Albert-Ludwigs-Universität Freiburg, 79110 Freiburg, Germany

Abstract. The design of distributed, safety critical real-time systems is challenging due to their high complexity, the potentially large number of components, and complicated requirements and environment assumptions. Our case study shows that despite those challenges, the automated formal verification of such systems is not only possible, but practicable even in the context of small to medium-sized enterprises. We considered a wireless fire alarm system and uncovered severe design flaws. For an improved design, we provided dependable verification results which in particular ensure that conformance tests for a relevant regulation standard will be passed. In general we observe that if system tests are specified by generalized test procedures, then *verifying* that a system will pass any test following these test procedures is a cost-efficient approach to improve product quality based on formal methods.

1 Introduction

Wireless communication offers a low-cost solution for distributed sensing and actuation systems. In recent years, wireless systems have expanded their roles towards performing an increasing number of safety critical tasks. The addition of more features inevitably increases their complexity and with it the risk of critical malfunctions. Consequently, there is a pressing need for methods and tools to verify the safety of wireless systems in critical applications. In this paper, we report on the verification of a wireless fire alarm system. Wireless fire alarm systems are increasingly preferred over wired ones due to advantages such as, e.g., spatial flexibility.

The main purpose of a fire alarm system is to reliably and timely notify occupants about the presence of indications for fire, such as smoke or high temperature. As system components may fail, e.g. due to physical damage, this purpose can in general not be guaranteed. Thus fire alarm systems need to employ self-monitoring procedures and notify maintainers if they are not able to fulfill their main purpose. Both false alarm notifications and false maintainer notifications should be avoided as they induce unnecessary costs.

Given the safety and liability issues associated with system failures, it is necessary to establish, with a good level of confidence, that the system design is

correct with respect to its requirements. In our case study, we accompanied the development of a wireless fire alarm system (WFAS) by SeCa GmbH [2], a small company specialized in radio technology. We consider requirements from the European standard EN-54, part 25 [8], which regulates the main obligations for commercially available WFAS. These requirements are stated as test procedure specifications. For example, the triggering of a single sensor anywhere in the system must cause a fire notification within 10 seconds. We generalized and formalized these test procedures and verified that the WFAS design will pass EN-54 conformance tests executed according to those test procedures in all possible scenarios.

Given the characteristics of the system, conventional testing poses considerable technical challenges: It is difficult to precisely control environment conditions such as radio interference and the timing of system inputs, not to mention the need for a prototype implementation and hardware. We propose the use of formal verification tools and techniques to overcome these difficulties. Nonetheless, state-of-the-art verification techniques also face several challenges while treating a system with such characteristics. First, the number of components and topologies is large. Second, the standard documents explicitly specify complex environment assumptions which need to be considered in the analysis. Third, the relevant properties are real-time properties. A further challenge was posed by the fact that we analyzed the design a priori, i.e. during its development, thus it was necessary to efficiently handle design changes.

The primary goal of our case study was to ensure that EN-54 certification tests will not fail due to design flaws. To this end we needed to provide the company with sufficient evidence that the system fulfills its requirements while giving a detailed account of the assumptions and limitations of the analysis, i.e. we needed to provide dependable [15] analysis results. An additional goal was to study the feasibility of applying formal methods to the verification problems found in small to medium-sized enterprises (SMEs).

During our case study we assumed the role of consultants and became an active component of the development team of the company. We created, validated, and verified models of the WFAS under design and the environment conditions specified by EN-54 25, using several formalisms and tools. Most aspects of the protocol are modelled using timed automata [3], which were subsequently verified using Uppaal [4]. Untimed liveness aspects of the alarm functionality of the system were verified using SPIN [14].

Several case studies on the verification of safety critical and distributed systems have been published. Fehnker et al. [10] report on verification of the ad hoc on-demand distance vector routing protocol (AODV) where they enumerate all possible topologies with up to 5 nodes. We provide verification results for a wider range of topologies. Closer to our work is the verification of an AODV protocol draft [5]. Their approach using HOL and SPIN considers real-time only in form of integral factors, and they did not validate the models with the development engineers. Other works [21,17] report on verification of the CAN bus protocol and a self-recovery algorithm without considering real-time.

Madl et al. [19] use UPPAAL as part of a model-based verification framework that performs limited compositional analysis where only isolated aspects of the considered protocol are modeled using timed automata. Semi-automatic verification of time-triggered architectures has also been carried out in the works of Kopetz et al. [18] and Tripakis et al. [23]. Our verification strategy is based on the use of fully automatic tools to discharge the main requirements after decomposition and optimization. In addition, verification of real-time properties of wireless sensor networks found in the literature is typically applied *a posteriori*, e.g., for LUNAR [24], an implementation already existed, and they consider to "employ a formal construction method rather than a post-construction verification" as future work. Similarly, Dong et al. [9] verified E-2C when it was already "in test" and Gebremichael [12] verifies the then well-known Zeroconf protocol.

The next section describes the requirements analyzed in our case study together with our formalization. Thereafter, Sections 3 and 4, describe the verification of a concrete system design which aims at implementing those requirements.

2 Requirements

Fire alarm systems are regulated in the European Union by the standard EN 54 [8]. Its requirements are expressed with respect to *components*, e.g. sensors, and a *central unit*. The central unit is a special device which is the interface of the system with its users. It in particular *displays* events such as alarms and sensor failures. Indications of fire detected by the *sensors* raise alarm events. Sensors communicate using radio channels to cause the central unit to display an alarm. Sensors need to be monitored constantly in order to ensure that their communication path towards the central unit is functioning correctly. The detection of a sensor failure is also required to be displayed at the central unit.

2.1 Formalization of EN 54-25 Requirements

For each requirement, Part 25 of the EN 54 standard contains specifications of test procedures including environment conditions and test engineer interactions. All test procedures assume a *ready-for-use system* as specified by the system manufacturer. A system is ready-for-use when the alarm and monitoring functions are fully operational and events are expected to be detected and displayed. During a test, one can assume that the test engineer conducts exactly those interactions with the system that are prescribed by a particular test procedure. Additionally, the standard specifies the number of system components that should be used for each test. There are certification authorities which perform tests based on the procedures specified by the standard and issue compliance certificates if those tests are passed. Thus we consider a design *correct* if it is guaranteed to pass all tests following those test procedures.

The standard provides test procedures for the following requirements:

1. The loss of the ability of the system to transmit a signal from a component to the central unit is detected in less than 300 seconds and displayed at the central unit within 100 seconds thereafter.

$$\bigwedge_{i\in C} \neg\Diamond(\lceil FAIL = i\rceil \,;\, \lceil FAIL \neq i\rceil) \quad (\mathsf{FailPers}_T) \qquad \neg\bigvee_{i\in C}\lceil FAIL = i\rceil \quad (\mathsf{NoFail}_T)$$

$$\Box\big[\neg\big(\bigvee_{j,k\in F, j\neq k}\lceil JAM_j \wedge JAM_k\rceil\big) \wedge \big(\lceil\bigwedge_{j\in F}\neg JAM_j\rceil \implies \ell \leq 1s\big) \wedge \bigwedge_{j\in F}\big(\lceil\neg JAM_j\rceil \,;\, \lceil JAM_j\rceil \,;\, \lceil\neg JAM_j\rceil \implies \ell \geq 1s\big)\big] \quad (\mathsf{Jam}_T)$$

$$\neg\bigvee_{i\in C}\lceil AL_i\rceil \quad (\mathsf{NoAl}_T) \qquad \bigwedge_{i\in C}\Box\,(\lceil DET_i\rceil \implies \lceil FAIL = i\rceil) \quad (\mathsf{NoSpur}_T)$$

$$\bigwedge_{i\in C}\Box\,(\lceil FAIL = i \wedge \neg DET_i\rceil \implies \ell \leq 300\text{s}) \quad (\mathsf{Detect}_T)$$

$$\bigwedge_{i\in C}\Box\,(\lceil DET_i \wedge \neg DISP_i\rceil \implies \ell \leq 100\text{s}) \quad (\mathsf{Display}_T)$$

$$\mathsf{FailPers}_T \wedge \mathsf{Jam}_T \wedge \mathsf{NoAl}_T \Rightarrow \Box\,(\lceil RDY\rceil \Rightarrow \mathsf{Detect}_T \wedge \mathsf{Display}_T \wedge \mathsf{NoSpur}_T) \quad (\mathsf{Req1}_T)$$

$$\mathsf{Jam}_T \wedge \mathsf{NoFail}_T \implies \Box\,(\lceil RDY\rceil \implies \mathsf{Alarm1}_T \wedge \mathsf{Alarm2}_T \wedge \mathsf{Alarm10}_T) \quad (\mathsf{Req2}_T)$$

Table 1. DC formalization of the considered requirements.

2. A single alarm event is displayed at the central unit within 10 seconds.
3. Two alarm events occurring within 2 seconds of each other are each displayed at the central unit within 10 seconds after their occurrence.
4. Out of exactly ten alarms occurring simultaneously, the first should be displayed at the central unit within 10 seconds and all others within 100 seconds.
5. There must be no spurious displays of events at the central unit.
6. Requirements 1 to 5 must hold as well in the presence of radio interference by other users of the frequency band. Radio interference by other users of the frequency band is simulated by a jamming device specified in the standard.

We already provided a formalization of the test procedures in [7]. From that, we derived testable Duration Calculus (DC) [6] properties. We call a component responsible for monitoring another a *master*, the monitored component is called a *slave*. The master-slave relation of a system is called its *topology*. Let T be a WFAS topology with the finite set C of components in addition to the central unit which use the frequency bands (or radio channels) in the set F. We assume the following observables for T. For $i \in C, j \in F$:

- RDY: *true* iff the system has been declared ready for use.
- $FAIL$: i iff component i is unable to transmit to the central unit, $\perp$ otherwise.
- DET_i: *true* iff the master of component i has detected a failure at i.
- $DISP_i$: *true* iff the central unit has displayed an event at component i.
- AL_i: *true* iff component i has detected an event.
- JAM_j: *true* iff radio channel j is being jammed.

The standard specifies that at most one component may be disabled during the test, and that disabled components are never re-enabled ($\mathsf{FailPers}_T$, cf. Table 1). That is, for each component, there is no interval which can be chopped into one phase where the component is disabled followed by a second phase where the component is not disabled.

The jamming devices used during certification tests are specified as (i) only the radio channels used by the system under test are jammed, (ii) only one, non-deterministically selected radio channel is jammed at a time, (iii) channels are continuously jammed for at least one second, and (iv) during channel changes,

all radio channels are free for at most one second. More formally, we obtain Jam_T (cf. Table 1). Note that it is especially hard for conventional testing approaches to cover the non-deterministic behaviour of the jamming device.

The standard states that during monitoring testing, the system is considered to be free of alarms (NoAl_T) and that during alarm testing, failures are not considered (NoFail_T). Requirement 5 and the deadlines of Requirement 1 can be formalized by NoSpur_T, Detect_T, and $\mathsf{Display}_T$. The complete, formal requirement for the monitoring function is $\mathsf{Req1}_T$. Similarly, we formalized Requirements 2, 3, and 4 as $\mathsf{Alarm1}_T$, $\mathsf{Alarm2}_T$, and $\mathsf{Alarm10}_T$. Overall, we have $\mathsf{Req2}_T$ for the alarm function. Note that $\mathsf{Req1}_T$ and $\mathsf{Req2}_T$ include Jam_T to express Requirement 6. Requirements 1 to 5 without requirement 6 can be formulated by redefining Jam_T to $\Box\lceil\bigwedge_{j\in F}\neg JAM_j\rceil$. We have thus formalized all requirements necessary to formally verify protocol designs against the standard.

3 Verification of the Monitoring Function

The WFAS under design is expected to work in a broad variety of buildings with possibly suboptimal conditions for radio signals. Therefore, the developers employ *repeaters* to relay messages in addition to the mandatory central unit and the sensors. In the WFAS topology, each component is assigned a unique master. The master-slave relation forms a tree with the central unit as root. Figure 1 depicts an instance of a WFAS topology with sensors $S_1, \ldots, S_6$, repeaters $R_1, \ldots, R_3$, and the central unit CU. Repeaters and the central unit function as master i and consist of two transceivers each, Tr_1^i and Tr_2^i. For displaying an event, a repeater notifies its master of the incidence, which notifies its master until the notification reaches the central unit. Functions of the protocol are distributed among the two transceivers in the masters. Transceiver Tr_1^i is only used for sensor monitoring, that is, this transceiver realizes the master-role towards sensors. Transceiver Tr_2^i realizes three functions: the slave-role towards another repeater or the central unit, the master-role towards other repeaters, and the forwarding of events.

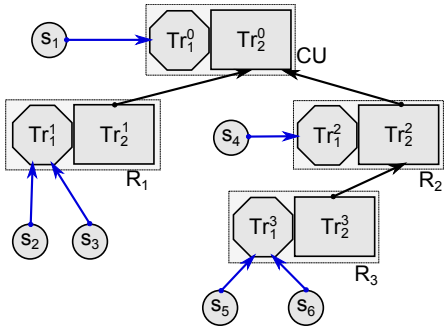


Fig. 1. Example of a system topology.

The protocol designed employs a variant of Time Division Multiple Access (TDMA) as shown in Figure 2. Time is partitioned into frames and frames are divided into fixed-width windows. Windows are in turn subdivided into slots, which are assigned to different protocol functions. The window length is specified in *tics*. Every sensor and repeater is assigned a unique window.

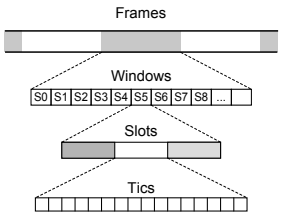


Fig. 2. TDMA scheme.

To perform failure detection, repeaters in the slave-role and sensors use the same functionality. Slaves periodically send a *keep-alive* message to their master in the corresponding slot of their assigned window. If no *acknowledge* message is received from the master, a second and third *keep-alive* are transmitted in the subsequent slots using a different channel. Masters listen on the corresponding channel during the slots of their assigned slaves. A master enters its error detection status when a specified number of keep-alive messages from one slave have consecutively been missed. The master then initiates the forwarding of the failure detection event. Event forwarding takes place without regarding slot assignments, using the transceivers Tr_2^i.

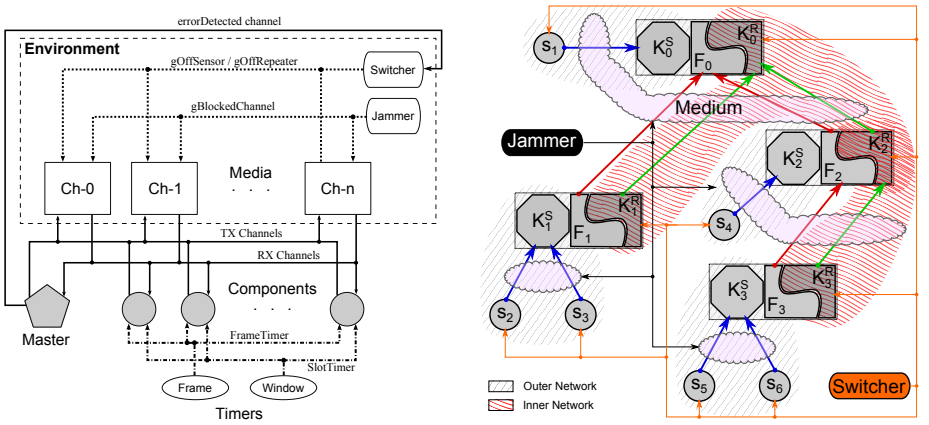
To compensate for unavoidable clock drift, i.e., slight deviations between clock speed in different components, a correction mechanism is used. Acknowledgements for keep-alive messages come with a time stamp which allows the slave to synchronize its clock with the master's clock. Additional time intervals added at the beginning and at the end of slots (*guard times*) ensure that transmissions of keep-alive messages do not overlap and are not lost. In the design, sensors stop sending keep-alive messages after a determined number of consecutive non-acknowledged keep-alive messages because they are then missing a sufficiently recent time stamp. This mechanism prevents a malfunctioning sensor from causing message collisions.

3.1 Modeling

The requirements (cf. Section 2.1) indicate a clear separation between environment assumptions and protocol components. We employ a reusable, modular environment model which is coupled to a protocol design model by a defined interface (cf. Figure 3(a)). We can thus accommodate changing design ideas during development while maintaining fixed environment assumptions. A sample topology including all necessary model artifacts is shown in Figure 3(b). In the following, we present our environment model and our model of the final design of the monitoring function.

Environment Model. The environment consists of modules that represent radio channels (Media), non-deterministic component failures (Switcher), and radio interference by a jamming device (Jammer). To allow parallel communication over different radio channels, the communication medium consists of one timed automaton for each channel used by the system. System models send messages to the media using a synchronisation channel array `TX`, indexed with the message type and the channel used. A medium then broadcasts the message using the `RX` channel array with the same indexing conventions. When a component is deactivated by the Switcher automaton, any message sent by the component is discarded at the media without being relayed. Likewise, messages are discarded at the radio channel blocked by Jammer.

The protocol is designed to be free of collisions. To verify this property, media models were designed to accept only one message at a time. If two components



(a) Model architecture. (b) Inner and outer networks (cf. Sect. 3.2).

Fig. 3. Model architecture and exemplary model instance for the monitoring function.

send messages simultaneously, a *deadlock* occurs. Verifying the absence of collisions thus amounts to checking whether the complete model has deadlocks.

Radio interference is modeled by the Jammer automaton (see Figure 4). The currently blocked radio channel is indicated by the global variable `gBlockedChannel`. If all radio channels are free, its value is -1. As the radio jammer may have been switched on before the system is ready for use, blocking of a radio channel as observed by the ready-for-use system may be initially *shorter* than 1s (cf. Jam_T on page 4). This situation is explicitly modeled by location *INIT_BLOCKED*.

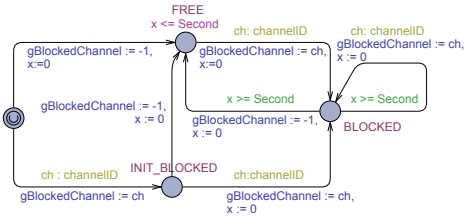


Fig. 4. Radio jammer model.

Clock reduction. Motivated by the large number of components, we applied *quasi-equal clock reduction* [13]. This technique takes advantage of clock variables that have the same value except for discrete points in time. These clocks are reduced to a single, centralized clock and thus verification complexity is decreased. In our case, the environment model includes the central clock sources Frame and Window. They provide a global clock variable which is reset at the end of each window and use a broadcast channel to notify components whenever the clock is reset. The automaton Window also handles keeping track of the number of windows passed since the beginning of the frame. Note that symmetry reduction can not be applied because the assignment of slots to components is based on the components' identity.

To make the simplifying assumption that clocks are perfectly synchronized and can be reduced, we verified separately that the guard times used in the design satisfy the conditions of [16]. These conditions ensure that keep-alive messages do not collide given that quartz oscillators work inside their specified ranges.

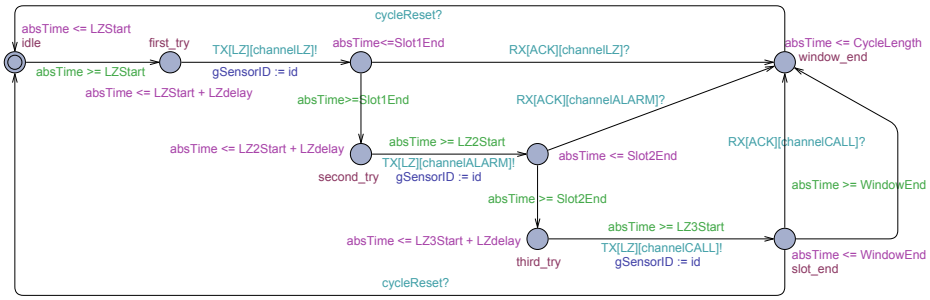


Fig. 5. Model of the slave-role of components.

Monitoring Protocol Model. The protocol design is modeled by the different system components that perform the functions of masters and slaves. Figure 5 shows the timed automaton modelling the slave-role of sensors. Note that the keep-alive message is potentially sent three times in a slot.

A repeater is basically modelled by three sub-models: the master-role towards sensors, the master-role towards repeaters, and the slave-role of repeaters. A master-role is further subdivided into two sub-models: the first one receives keep-alives and replies with acknowledgements, the second one keeps track of missing messages to determine the status of its slaves. In order to observe DET_i in the model, the second master-role timed automaton synchronizes with the Switcher automaton once an error is detected. A central unit is similar to a repeater, just without the model for the slave-role.

In order to maintain simple and readable models, assumptions of the main requirement $\mathsf{Req1}_T$ are integrated directly into the model. For instance, the fact that RDY holds permanently is modeled by the fact that no operation modes outside of a ready-for-use system are modeled. Likewise, no alarms or their corresponding features are present in the model in order to satisfy NoAl_T. The persistence of failures, $\mathsf{FailPers}_T$, is modeled directly by not allowing the timed automata to return to normal operation once they have been deactivated. Additionally, the Switcher automaton allows for only one component to be deactivated during a system run (cf. Section 2.1).

3.2 Verification

We realized the environment and protocol model including assumption treatment in Uppaal. In the models [1], the observables from Section 2.1 are modeled either as locations in the timed automata, or as mappings to (both continuous and discrete) variable values. Uppaal queries can be derived from the *testable* [20] DC formulae in Section 2.1 to check the satisfaction of the requirements.

Decomposition: Inner and Outer Network. As the two transceivers operate on different radio channels, any WFAS topology can be seen as consisting of two independent networks with the repeaters as gateways. We call the network used for communication between repeaters and the central unit the *inner* network

Query	Sensors as slaves, $N = 126$. seconds	MB	States	Repeaters as slaves, $N = 10$. seconds	MB	States
Detect_T	36,070.78	3,419.00	190,582,600	231.84	230.59	6,009,120
NoSpur_T	97.44	44.29	640,943	3.94	10.14	144,613
No LZ-Collision	12,895.17	2,343.00	68,022,052	368.58	250.91	9,600,062
Detection Possible	10,205.13	557.00	26,445,788	38.21	55.67	1,250,596

Table 2. Verification of the final design (Opteron 6174 2.2Ghz, 64GB, UPPAAL 4.1.3).

and the network used to communicate between sensors and repeaters the *outer* network. In Figure 3(b), instances of inner and outer networks are highlighted for the depicted topology.

The detection aspect of the monitoring functionality, Detect_T, takes place *local* to an inner or outer network. Thus it can be verified by regarding a master and its set of slaves in isolation, since monitoring subnetworks do not communicate or interfere with each other because of the TDMA scheduling. We thus verified separate models for the monitoring function of sensors (outer network) and of repeaters (inner network), while abstracting away the networks and components that do not participate in the detection function.

Topology Coverage. We performed verification on two models, each consisting of a master with the maximum number of slaves allowed by the design as required by EN 54-25. The sensor to master model comprises a master (representing both a repeater and the central unit) and 126 sensors, plus the environment models (Jammer, Switcher, and Media). Verifying a subtopology with the maximum number of connected sensors subsumes all other subtopologies with a smaller number of components because a functioning sensor overapproximates, in particular, the behavior of an absent sensor. Together with the observation that subnetworks are isolated in their detection function, verifying detection on the model provided is sufficient to prove the satisfaction of Detect_T and NoSpur_T in all topologies T, when each topology is considered as a collection of isolated outer network subtopologies. Analogously, the model of the master role towards repeaters, which contains 10 repeaters, covers all topologies.

Results. The verification of a model of the initial design of the monitoring protocol resulted in the discovery of two design faults: A corner case in which the detection deadline was exceeded by one tic and a violation of NoSpur_T in the presence of a jamming signal. Given the timing constraints specified for the jamming signal, it was possible for it to continuously block the keep-alive messages of a sensor, thus causing spurious failures to be detected at its master. The design was consequently adapted to shorten the length of the windows assigned to sensors and to include an additional retry (to make three in total) for the transmission of the keep-alive packets, together with a faster channel rotation scheme. Verification of the adapted design revealed no further vulnerabilities (cf. Table 2). Likewise, we verified NoSpur_T in the final model and performed

additional checks to establish validity: Reachability of the detection location excludes trivial satisfaction of the requirements, and the absence of deadlocks excluded message collisions. In addition, we verified that Jammer satisfies Jam_T.

Having successfully verified the detection phase of the protocol, we still need to verify $\mathsf{Display}_T$ in order to discharge requirement $\mathsf{Req1}_T$. As forwarded failures and alarms are treated equally according to the design, we condition the satisfaction of $\mathsf{Display}_T$ to the satisfaction of the alarm deadline requirements. Given that the deadline for displaying failures is much larger (100 seconds as opposed to 10 seconds for alarms), we deduce that satisfying alarm deadlines subsumes satisfying failure display deadlines.

4 Alarm Verification

Whenever an event is detected, it is forwarded towards the central unit to be displayed. Forwarding is performed by the second transceiver Tr_2, which always uses a different radio channel than Tr_1. An event is transmitted in the slot immediately following detection or reception for forwarding, thus ignoring the window assignments. This design choice speeds up transmission, but allows for *collisions*, i.e. simultaneous transmission of two or more messages. When a collision occurs, the information of the participating messages may be distorted or destroyed.

A resolution protocol was devised in order to accommodate for the possibility of collisions. The protocol follows a tree-splitting [11] approach based on the system-wide unique ID numbers of the components. The protocol assigns a tic for the start point of the transmission in the next slot. At the start point, a component listens shortly to determine whether the channel is free, in which case the transmission is started ("listen before talk"). If the channel is in use, the component waits until the next slot to retry transmitting using the same starting point. After a transmission, if no acknowledgment is received, a collision is assumed and, based on the binary representation of the ID and the respective collision counter, the colliding components deterministically modify their starting point for retrying the transmission in the next slot. The process is repeated as long as an event is not successfully transmitted and should guarantee that every waiting event eventually reaches the central unit.

In Figure 6, the initial steps of the collision resolution are depicted for an example scenario. Assume, components with IDs 127, 85, 42 and 1 detect an event at time t_0. In the slot following t_0, all components start transmitting their events at initial transmission start tic 8, which results in a collision of all 4 messages. All components detect the situation (they did not receive an acknowledgement),

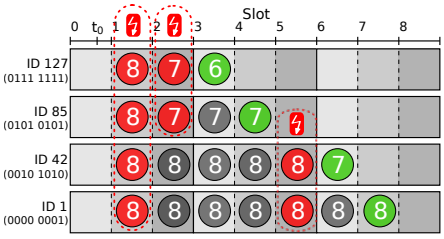


Fig. 6. Collision resolution. The transmission start tics are shown inside the circles. Colliding components are shown in red, waiting ones in gray, and successful transmissions in green.

advance their collision counter, and choose based on their ID and the new value of the collision counter a new starting point. In slot 2, components 127 and 85 start their transmission at tic 7 while components 42 and 1 continue to start at tic 8. The messages of components 127 and 85 collide once again, while 42 and 1 find the channel in use and wait for the next slot to retry. In slot 3, all colliding components advance their counter and now 127 chooses an earlier starting point while 85's starting point does not change. This allows 127 to transmit successfully while the remaining components detect the transmission and wait. The process repeats and 85, 42 and 1 finally deliver their events to their masters.

4.1 Alarm Deadlines

Environment and Collision Resolution Model. For the environment, we employed an architecture similar to the one described in Section 3. The media models were extended to explicitly accept and observe message collisions. A newly added parameter models the range of radio communications, which limits the number of components whose messages may collide. With a value of 1, only components connected to the same master, and their master can collide, higher values allow collisions with components connected to masters further away.

Additionally, a simpler radio jammer was used. The non-deterministic jammer as shown in Figure 4 proved too complex in its behavior, causing the verification to timeout. After consulting with the company and the certification authority, we elicited additional assumptions about the jamming device used for EN-54 certification. Those assumptions allowed us to model and use a *sequential* jammer, that deterministically chooses the channel to be jammed. Again, we used quasi-equal clock reduction and the assumption that clocks are perfectly synchronized. Note that, due to listen before talk, message collisions only occur if transmissions start at the exact same time, hence perfectly synchronized clocks present a more difficult scenario for collisions.

We modelled the tree splitting collision resolution algorithm for the alarm behavior of the sensors and the forwarding component of repeaters. Repeaters employ an event queue implemented as an array with a pointer variable. For the verification of a single alarm and ten simultaneous alarms, all sensors start in the alarmed state, this without loss of generality since it is the common behavior for all possible detection time points in the previous slot. In the model for two alarms, non-determinism is introduced to allow for the alarms to occur at all possible points inside a 2-second interval, in particular simultaneously.

Verification and Results. The event forwarding mechanism of the proposed design posed a challenge for verification for two main reasons: (a) The forwarding times strongly depend on the topology, in particular on the number of repeaters along the way to the central unit, and (b) the algorithm employs complex data structures.

The EN-54 standard requires that an "especially difficult" topology is used in certification tests. The developers agreed with the certificate authority on

		$T = T_1$ (palm tree, full coll.)			$T = T_2$ (palm tree, limited coll.)		
Query	**ids**	**seconds**	**MB**	**States**	**seconds**	**MB**	**States**
$\mathsf{Alarm1}_T$	-	3.6 ± 1	43.1 ± 1	$59k \pm 15k$	1.4 ± 1	38.3 ± 1	$36k \pm 14k$
$\mathsf{Alarm2}_T$	seq	4.7	67.1	110,207	0.5	24.1	19,528
$\mathsf{Alarm10}_T$	seq	44.6 ± 11	311.4 ± 102	$641k \pm 159k$	17.3 ± 6	179.1 ± 61	$419k \pm 124k$
	opt	41.8 ± 10	306.6 ± 80	$600k \pm 140k$	17.1 ± 6	182.2 ± 64	$412k \pm 124k$

Table 3. Resource consumption for the verification of the alarm functionality.

one which involves the maximum number of chained repeaters allowed by the design and is expected to cause many collisions based on the IDs of the involved components. This topology resembles a "palm tree", with the central unit as root, 5 chained repeaters, and 10 sensors connected to the farthest repeater.

We realized the chosen topology using Uppaal, but had to use the convex-hull overapproximation [25] to successfully verify all properties. For the verification runs, we considered different scenarios for the "palm tree" topology along several dimensions: 1. The range of the collisions ("full" for all components colliding, "limited" for only neighboring components colliding) 2. The assignment of IDs for the colliding sensors ("opt" for optimized IDs with large edit distances and "seq" for IDs sequentially assigned) 3. Which receiver is influenced by the radio jammer (averaged results are shown). Average time, memory and states explored are shown in Table 3. Additionally, we were requested by the company to extract the expected worst-case response times for alarm delivery. We employed the `inf` and `sup` functions provided by Uppaal, and obtained upper bounds for the time needed to deliver 10 simultaneous alarms with the IDs sets we considered: in T_1 the 1st alarm is displayed after at most 4.32s (T_2: 5.89s), and the 10th alarm after at most 44.4s (T_2: 33.45s). As soon as prototype hardware and software were available, the developers measured the response times for different scenarios and proved the model predictions accurate. This validation step enhanced the confidence of the developers in their design.

4.2 Non-Starvation of the Collision Resolution

Although valuable for certification, only limited topology and scenario coverages are achieved by the results reported on in Section 4.1. To increase confidence on the effectiveness of the collision resolution protocol, we set out to provide evidence that delivery of messages is ensured. That is, that the protocol satisfies the liveness property that a message delivery request is *eventually* served.

Untimed Collision Resolution Model. For more general scenarios than the ones considered in Section 4.1, Uppaal proved unwieldy. Thus we provide a Promela [14] model for SPIN, which is a state-of-the-art tool for checking models with bounded integer data. In our model, a single process non-deterministically selects, from a given set I of component IDs, $N \leq |I|$ component IDs which will detect an alarm. The protocol state is encoded by an array indexed by the component IDs allowed in the system (256 in our case). At each position,

there is a collision counter and binary flags indicating whether the component detected an alarm and whether it has delivered its message. One step of the model represents the evolution during one slot. In the step, collisions are detected and start-times are updated by executing the collision resolution algorithm for each active entry of the array. The time *when* the alarm is detected is also chosen non-deterministically for each selected component. Thus with, e.g., a given set $I = \{i_1, \ldots, i_{10}\}$ of 10 IDs the case $N = 3$ analyses all possible collisions of size 1, 2, and 3 with any possible overlap in time. For instance, the case that only i_1, i_2 detect an alarm at the same time is covered by the case where i_3 detects an alarm earlier and immediately transmits its message successfully.

Topology Coverage. The model represents one-hop collisions, that is, collisions between sets of components whose messages can collide, similar to the models employed for the verification of the failure detection mechanism (cf. Section 3.1). A choice of N IDs from I in our model covers all topologies where those IDs are logically and physically distributed such that their messages may collide. Verifying the model for a given value of N is equivalent to checking liveness for the protocol in all topologies with up to N colliding messages.

Verification and Results. The analysis of the untimed model uncovered several vulnerabilities of the protocol. Firstly, there is an issue present in all carrier sensing protocols: The hidden terminal problem. When two components are unable to detect the transmissions of each other and repeatedly transmit simultaneously, effectively causing a common receiver to lose all information. The problem was deemed, however, acceptable by the developers, since it rarely occurs in practice and can be easily avoided by slightly adapting the physical distribution of sensors. Therefore we only considered scenarios without the hidden terminal problem for the verification.

$\|I\|$	N	**sec.**	**MB**	**States**
255	2	49	1,610	1,235,970
H	10	3,393	6,390	6,242,610
L	10	4,271	10,685	10,439,545
Rnd	10	4,465	11,534	11,268,368
average		4,138	9,994	9,763,809

Table 4. Resource consumption for the verification with SPIN 6.2.3.

For component selections with $N = 2$, a problem of the limited number of starting points for transmissions was uncovered: Whenever components with IDs 0 and 128 entered into collision resolution with a third component causing them to collide at the same tic, the algorithm caused the collision to repeat in an endless loop. Due to the similar binary representations of the IDs, identical and repeating start point selections were made by both components. The company adapted the configuration tools for the WFAS in a way that avoids assigning these IDs to components in close physical proximity. For higher values of N, the memory and time available were insufficient. We thus resorted to sampling IDs according to the similarity of their binary representation. We observe that the steps of the collision resolution algorithm correlate with the similarity between the binary strings of the IDs The average resource consumption figures for our verification effort with SPIN are shown in Table 4. We performed random selec-

tion of ID assignments for collisions of 10 sensors. The sampling was guided to explore the effect of similarity using the *Hamming distance* of the components within a sample. Three different categories were chosen: low similarity (L), high similarity (H), and pure random samples (Rnd). In total, we sampled 31,744 different 10-component selections. As seen in Table 4, similarity appears to have an inverse correlation with the size of the explored state space. We can thus assume that a good coverage over the space of ID selections was achieved.

5 Conclusion

The formal modelling and verification of the new Wireless Fire Alarm System proved challenging. Employing different techniques and tools such as property decomposition, internal assumption treatment, meta-reasoning about topology coverage, and timed and untimed verification support of Uppaal and SPIN enabled us to dependably provide sufficient evidence that EN-54 certification tests will not fail due to design flaws. All models are available for download [1].

Our verification effort proved valuable for the development process of the company involved. We discovered previously unknown flaws that triggered significant revisions of the design. For the final design, we delivered concrete information about the operational circumstances for which our verification results apply. According to the testimony of the company, the project was accelerated compared to previous developments without the use of formal methods: The first prototype implementation already passed all initial in-house tests, thus the test phase was substantially shortened and the effort of bug-fixing ameliorated.

What can be learned from our effort? We feel that the key for providing valuable results in limited time was to generalize and formally specify relevant and involved *test procedures*, and verify that tests following those will be passed.

Because most companies specify tests and are used to this activity, formalizing test procedures seems to be a cost-efficient way of obtaining precise specifications for formal verification. From our experience, verifying that a design will pass all tests according to the given generalized test procedure can avoid huge test efforts; in addition, verification does not require initial implementations and hardware prototypes as opposed to conventional testing.

We related the models to the knowledge and experience of the designers by simulating different scenarios directly in Uppaal. Discussing these scenarios facilitated the assessments of whether the models faithfully represent the design under development, i.e. model validation. In our case, a further indication for validity is that time bounds predicted by the model could be confirmed by the developers by measuring the implemented system.

Of course, the development goal here was not a system which only passes the certification tests, but a *good system*, one that fulfills all functions it was designed for. The WFAS, for example, should properly handle the failure of more than one sensor at a time. Checking such scenarios is possible with our techniques and would further increase confidence, but incurs additional costs for specification

and modelling. Here, conventional testing is appropriate: From knowledge about the models, we expect the given scenarios to pass.

We see that formal methods and tools available today are capable of treating problems of SMEs while adding value to their development process. *A priori* design verification as conducted in our case study facilitates finding design errors early and potentially saving efforts and costs. For the certification of critical systems, verification of design models could also improve certification processes, in addition to the verification of binaries as in DO-333 [22].

References

1. http://swt.informatik.uni-freiburg.de/projects/CaseStudyRepository/WFAS
2. SeCa GmbH, http://seca-online.de/home.html,313
3. Alur, R., Dill, D.: A theory of timed automata. TCS 126(2), 183 – 235 (1994)
4. Behrmann, G., David, A., Larsen, K.: A tutorial on Uppaal. In: SFM-RT 2004. pp. 200–236. No. 3185 in LNCS, Springer (2004)
5. Bhargavan, K., Obradovic, D., Gunter, C.A.: Formal verification of standards for distance vector routing protocols. J. ACM 49(4), 538–576 (Jul 2002)
6. Chaochen, Z., et al.: A calculus of durations. Inf. Proc. Lett. 40(5), 269–276 (1991)
7. Dietsch, D., Feo-Arenis, S., Westphal, B., et al.: Disambiguation of industrial standards through formalization and graphical languages. In: RE. pp. 265–270 (2011)
8. DIN e.V.: Fire detection and fire alarm systems; German version EN 54 (1997)
9. Dong, Y., Smolka, S.A., Stark, E.W., White, S.M.: Practical considerations in protocol verification: The e-2c case study. In: ICECCS. pp. 153– (1999)
10. Fehnker, A., et al.: Automated analysis of AODV using Uppaal. In: Flanagan, C., König, B. (eds.) TACAS. LNCS, vol. 7214, pp. 173–187. Springer (2012)
11. Garcés, R., Garcia-Luna-Aceves, J.J.: Collision avoidance and resolution multiple access (CARMA). Cluster Computing 1(2), 197–212 (1998)
12. Gebremichael, B., Vaandrager, F., Zhang, M.: Analysis of the Zeroconf protocol using Uppaal. In: EMSOFT. pp. 242–251. ACM (2006)
13. Herrera, C., et al.: Reducing quasi-equal clocks in networks of timed automata. In: Jurdzinski, M., et al. (eds.) FORMATS. LNCS, vol. 7595, pp. 155–170 (2012)
14. Holzmann, G.J.: The model checker SPIN. IEEE TSE 23(5), 279–295 (1997)
15. Jackson, D.: A Direct Path to Dependable Software. CACM 52(4), 78–88 (2009)
16. Jubran, O., Westphal, B.: Formal approach to guard time optimization for TDMA. In: RTNS. pp. 223–233. ACM (2013)
17. Kamali, M., et al.: Self-recovering sensor-actor networks. In: Mousavi, M.R., Salaün, G. (eds.) FOCLASA. EPTCS, vol. 30, pp. 47–61 (2010)
18. Kopetz, H., et al.: The time-triggered architecture. P. IEEE 91(1), 112–126 (2003)
19. Madl, G., et al.: Verifying distributed real-time properties of embedded systems via graph transformations and model checking. Real-Time Systems 33, 77–100 (2006)
20. Olderog, E.R., Dierks, H.: Real-time systems. Cambridge University Press (2008)
21. van Osch, M., et al.: Finite-state analysis of the CAN bus protocol. In: HASE (2001)
22. RTCA: DO-333 Formal Methods Supplement to DO-178C and DO-278A (2011)
23. Tripakis, S., et al.: Implementing synchronous models on loosely time triggered architectures. IEEE Transactions on Computers 57, 1300–1314 (2008)
24. Wibling, O., et al.: Automatized verification of ad hoc routing protocols. In: Frutos-Escrig, D., et al. (eds.) FORTE. LNCS, vol. 3235, pp. 343–358. Springer (2004)
25. Wong-Toi, H.: Symbolic Approximations for Verifying Real-Time Systems. Ph.D. thesis, Stanford University (1995)