A Case Study in Safety, Security, and Availability of Wireless-Enabled Aircraft Communication Networks*

Rohit Dureja†, Eric W. D. Rozier‡, and Kristin Y. Rozier§
Iowa State University, Ames, IA 50010, USA

As the costs of fuel and maintenance increase and regulations on weight and environmental impact tighten, there is an increasing push to transition on-board aircraft networks to wireless, reducing weight, fuel, maintenance time, and pollution. We outline a candidate short-range hybrid wired/wireless network for aircraft on-board communications using the common ZigBee protocol and privacy-preserving search implemented as a secure publish/subscribe system using specially coded meta-data. Formally specifying safety and security properties and modeling the network in nuXmv enables verification and fault analysis via model checking and lays the groundwork for future certification avenues. We report on our experiments building and testing our candidate hybrid network and report on overhead and availability for encrypted and fault-tolerant communications, and propose a system that allows system designers to directly trade fault-tolerance for bandwidth, or vice-versa, in an encrypted privacy-preserving framework.

I. Introduction

Increasing system complexity and limitations of wired networks in avionics systems are driving innovations in wireless technology integration. This migration of wired networks to wireless is motivated by several advantages: wireless networks weigh less than wired networks, scale better than comparable wired networks, and are easier to install and maintain since they are essentially plug-and-play. Although the cost of equipment for wireless networks are higher by comparison, these costs are typically outweighed by the benefits of ease-of-use, scalability, lower weight, and lower setup-cost than a comparable wired network. However, there are some draw-backs to migrating from wired to wireless networks. For example, wireless networks are more vulnerable to cyber-security attacks, providing a larger attack surface. Additionally, the added complexity in protocol and organization of wireless networks increases their susceptibility to faults. Since on-board networks in commercial aircraft are frequently implemented with multiple-redundancy, we consider *hybrid* wired-wireless networks where one or more of the redundant wires is replaced by a wireless connection. We report findings from our recent work designing next-generation aerospace standards for efficient, reconfigurable, aerospace systems. We examine the problem of selecting a wireless-enabled on-board network configuration along two dimensions: verification and formal fault analysis of a prototype hybrid network using an off-the-shelf protocol, and investigation of overhead and network availability for encrypted, fault-tolerant, protocols for such hybrid networks.

A modern Boeing 787 has ∼500km of wires; together with their harnesses they weigh nearly 7,400 kg, which is about 3% of the total weight of the aircraft.[1] An Airbus A380 also has ∼100,000 wires totaling 470 km and weighing 5,700 kg; another ∼1/3 of that weight is required for harnesses to hold these wires. Aerospace America describes this problem as the "War on Wiring," and estimates that up to 1,800kg of wiring could be removed from such aircraft, starting with non-avionics networks like passenger entertainment systems, system health management data networks, sensor networks, and networks carrying commands for flight-control.[1] Identifying wired components on the aircraft that can be migrated to wireless protocols while maintaining the robustness standards needed for flight certification remains an open question.

*Thanks to NASA's Efficient Reconfigurable Cockpit Design and Fleet Operations using Software Intensive, Networked and Wireless Enabled Architecture (ECON) Grant NNX15AQ84G for supporting this work.

†Graduate Student, Department of Computer Science, Iowa State University.
‡Assistant Professor, Department of Computer Science, Iowa State University.
§Assistant Professor, Deparments of Aerospace Engineering and Computer Science, Iowa State University.

Migration of wired networks to hybrid wired-wireless networks requires a thorough study of the wired system, choosing an appropriate wireless communication protocol, assessing faults and security concerns associated with the wireless communication, and analysis of the quantitative benefits of using a secured or unsecured wireless network. Each of these tasks are interdependent and require substantial understanding regarding expectations from the introduced wireless network with the primary requirement being that *the new, hybrid wired-wireless network needs to be at least as safe, and secure, as the existing wired network*. Our formal framework enables analysis of different hybrid wired-wireless network models, and allows us compare them to existing wired network models in terms of fault tolerance and fault propagation.

CONTRIBUTIONS. To the best of our knowledge, this is the first work that addresses the problem of communication technology migration in terms of system safety and fault tolerance. The formal framework we present aids system designers in the comparison of different communication networks and the exploration of viable fault-tolerant mechanisms. The presented framework builds upon existing model checking and safety assessment tools, and is plug-and-play, making it fully COTS compatible. As a proof of concept, we formally model the ZigBee protocol and demonstrate analysis of a hybrid network using ZigBee for its wireless protocol. We propose additions to the ZigBee protocol that enhance the reliability and trustworthiness of wireless communication, while ensuring real-time deadlines are met. The new format adheres to existing ZigBee standards, and can be used with COTS equipment. These modifications give the wireless system the ability to alter fault-tolerance and throughput capabilities in response to changing conditions on the aircraft.

RELATED WORK. The literature focuses on fault-tolerant mechanisms for ZigBee wireless sensor networks using existing wireless mechanisms such as Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA),[2] chained ZigBee repeaters, and the creation of multiple parallel transmission streams of the same data. These, and similar, mechanisms provide the basic tools for establishing safe ZigBee network topologies with the current state-of-the-art. In the case of hardware failure redundant networks of ZigBee transmitters have shown fault-tolerance using fault detection and recovery mechanisms through disconnection and network rejoin with altered node roles.[3] While these approaches have lead to more reliable systems, they do not account for the challenges of network migration, or hybrid networks in which the techniques are not necessarily fully supported. Wireless avionics, and human interfaces for spacecraft[4] are pushing towards migration of wired networks to wireless. In the absence of fault-tolerance mechanisms, experimental performance evaluation has been employed to assess network reliability[5] to understand the likelihood of failure when it cannot be prevented. However, such experimentation is limited in terms of scenarios explored and environmental conditions, and does not scale. Our framework focuses on inter-component communication rather than protocol behavior. It allows the comparison of different networks in terms of safety, and is fully automatic and scalable.

Algorithms and techniques for searchable encryption[6–8] have been studied extensively in the context of cryptography, but have focused mostly on traditional cryptographic applications, and the primary focus has been efficiency improvement and security formalization. The methods underlying searchable encryption[6] proved impractical and eroded security due to the necessity of generating every possible key that the search expression can match. To reduce the search cost, Bloom filters were used to create per-file searchable indexes[7] on the source data. However these studies required exact keyword search.

In theory, the use of flexible search over regular expressions is allowed[9] but again the results prove impractical, requiring cipher-text and token sizes of the order $O(2^{nw})$, where n is the number of input alphabet, and w is the the number of strings over the alphabet in the search query. Other recent studies have focused on cloud applications of data storage, such as the problem of similarity search[10] over the outsourced encrypted Cloud data and work that provides approximate search capability[11] for encrypted Cloud data. Approximate search capability is able to cover misspelling and typographical errors that exist in a search statement and in the source data, which significantly extends exact keyword search through adaptation of the metric space. Methods for encrypted data[12] allow the construction of a tree-based index that enables the retrieval of the relevant entries. With this indexing method, similarity queries can be carried out with a small number of evaluations.

Other studies in the literature make use of encryption techniques that ensure that user privacy is not compromised by a data center.[13] The challenge changes in these cases, becoming the problem of how an encrypted database can be queried without explicitly decrypting the records. At a high level, a searchable encryption scheme[14] provides a way to encrypt a search index such that its contents are hidden except to a party that is given appropriate tokens. Public Key Encryption with keyword search[15] indexes encrypted

documents using keywords. Public-key systems that support equality ($q = a$), comparison queries ($q > a$) as well as more general queries such as subset queries ($q \in S$) provide massive improvement for searchability. A symmetric cryptography setting for searchable encryption architectures[6] for equality tests advanced the state-of-the-art. Equality tests in the public-key setting are closely related to Anonymous Identity Based Encryption.[16] A new primitive called Hidden Vector Encryption[9] provides a highly performing and more general conjunctive query system allowing conjunction of equality tests, conjunction of comparison queries and subset queries.

II. Background

II.A. Model Checking

Model checking has been successfully used in the aviation industry to verify that avionics systems uphold their safety requirements.[17–29] We choose model checking as a verification method for our study due to the unique properties provided by this verification technique over using simulation or testing. These include the following:

- Model checking can analyze partial or incomplete designs; it can analyze both a logical system (such as code or a hardware logic design) and models of such a design as long as they are modeled in a formal semantics. This flexibility is required for our wireless network analysis, as we need to analyze a model of the partial design. (If safety requirements are violated by a partial design, adding more detail or implementation will not save the system.)
- Model checking is an exhaustive analysis of the entire behavior space of the system, for all possible inputs; it is not probabilistic.
- The exhaustive nature of model checking means that we can prove the *absence* of bad behaviors in addition to the presence of good behaviors; these two sides of the proof are required to construct an appropriate safety case for eventual flight certification, e.g., as required by DO-178B,[30] and DO-254.[31]
- Model checking finds the existence of any "bad" system execution, not how often that path is explored; in other words, for this type of analysis all bad paths are of equal weight. This is important for our early-design-time analysis because we want to be able to efficiently find violations of our safety requirements, however rare, so that we can proceed with a more complete knowledge of what could go wrong than we can obtain, e,g., from rare-event simulation.

Tools based on model-checking technology[32,33] have enjoyed a substantial and growing use over the last few years, and have recently been used to comparatively analyze multiple possible avionics system designs to narrow the design space early in the system design process.[20,21] Given a system or system model M in some formal semantics, and a requirement φ in some mathematical logic, model checking is the task of exhaustively and automatically checking whether the model satisfies the requirement, designated $M \vDash \varphi$. In the case that the system does not satisfy the requirement, the model checker returns a counterexample, which is a system execution trace exemplifying the requirement violation. The complete work flow is shown in Figure 1.

We use the symbolic model checking tool NuXmv[a] because it is well-documented,[35–37] freely available[b], and frequently used in industry.[26,38–47] However, there are several tools available with similar capabilities including CadenceSMV[48] (which has nearly identical syntax), SAL-SMC,[49] VIS,[50] and others. What sets NuXmv apart for our network design analysis is the ease of integrating an add-on for contract-based design, which we discuss in Section II.C.

II.B. Linear Temporal Logic

Because aerospace operational concepts are most frequently specified in terms of requirements over timelines, which represent possible scenarios, we choose a specification logic that naturally and intuitively encodes timelines: Linear Temporal Logic (LTL). LTL combines Boolean logic over system variables with how events unfold over time in different system scenarios. It is lightweight enough that model checkers can efficiently

[a] We are using NuXmv version 1.1.1 from `https://nuxmv.fbk.eu/`[34] for our experiments.

[b] The software and documentation are available for download from: `https://nuxmv.fbk.eu/`.

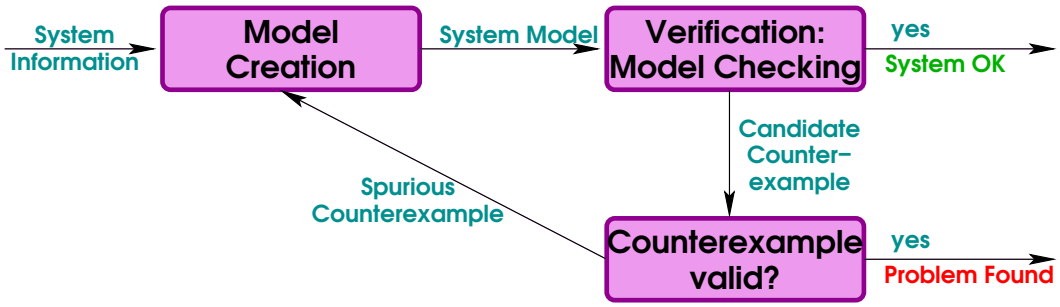
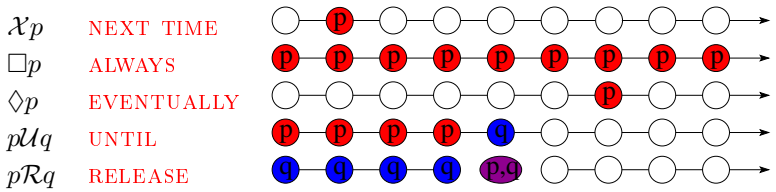


Figure 1: Model checking an early-stage, partial design, as we do in this study, involves entering the system information (as a model) and requirements into a model checking tool. If there is disagreement between the model's operation and its requirements, a *counterexample trace* is returned. Otherwise, the system satisfies the specification. The process is iterative, including validating the inputs by analyzing the outputs (counterexample or agreement) to ensure they make sense in terms of the real system.

analyze requirements specified in LTL, yet we found it to be expressive enough to specify the pertinent requirements of our on-board network designs.

Linear Temporal Logic (LTL) formulas reason about linear timelines; LTL requirements are formulas comprised of the following parts:

- finite set of atomic propositions {p q}
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$
- temporal connectives:



For a thorough survey on Linear Temporal Logic Symbolic Model Checking, with examples in NuXmv's modeling language, see.[51]

II.C. Contract-based Design

A *contract* is a clear and concise description of the expected behavior of a system. Contract-based design is an emerging paradigm for the design of complex, multi-component systems.[37,52–59] We associate each component in the system with a contract. A contract is an LTL formula of the form Always(*assumption* $\rightarrow$ *guarantee*). Contracts implement *assume-guarantee reasoning*. In a contract, *assumptions* are the expectations the component has from the environment in which it operates, and *guarantees* are behavioral promises fulfilled by the component provided the assumptions are met. Contract-based design allows for compositional modeling by breaking down a large system into smaller components. It facilitates efficient re-use of a component between several systems when the contract holds in changed environments; this also simplifies cross-validation of these models.

Figure 2a shows a system modeled compositionally. Each subcomponent in the system has an associated contract. The top-level system contract holds if and only if it is correctly refined by the contracts of its subcomponents. The contract-based methodology allows easy swapping of components as shown in Figure 2b. A library of components can be maintained and use in several designs. We use OCRA[53] for specifying contracts on the individual components of a wireless network.

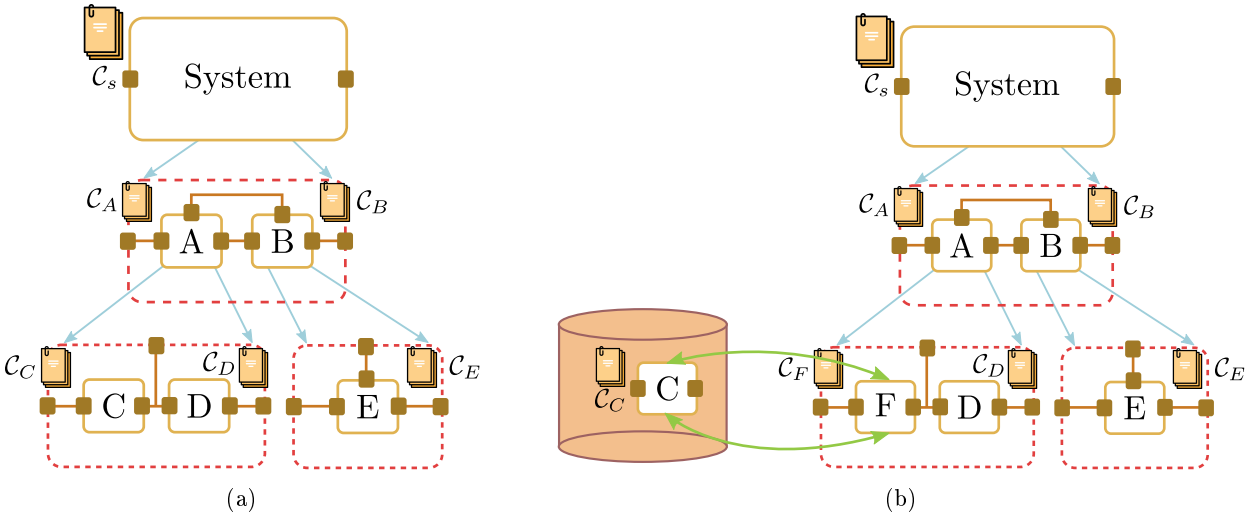


Figure 2: (a) Component hierarchy in a compositional system. The system is composed from components A and B. Component A is composed from component C and D, and component B from component E. (b) Contract-based design allows easy re-use of components from a library provided the contracts of the swapped components are the same.

II.D. Fault Tree Analysis

Fault-tree Analysis is a top-down deductive failure analysis technique that uses Boolean logic to combine a series of undesired lower-level events and their combinations that lead to overall system failure. We use fault tree analysis to compare different network models. Results of fault analysis can be reported in several ways: as fault trees, in tables, or in failure propagation graphs. Figure 3 shows how an undesired state of a system analyzed using Boolean logic to combine a series of lower-level events leading to the fault. The route through a tree between a fault and an initiating event is called a *Cut Set* and the shortest credible way through the tree from the fault to an initiating event is a *Minimal Cut Set*.

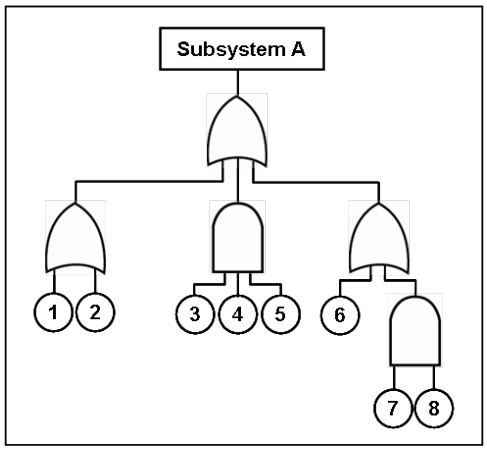


Figure 3: A sample fault-tree that uses Boolean logic to combine low level events that lead to an undesired state of a system

II.E. Fault-Tolerance in Data Systems

Reliability mechanisms for data and transmission have been extensively studied in the literature, and we leverage techniques from experimental and theoretical results on storage systems,[60–62] general system workloads,[63–65] application-specific I/O patterns,[66] and performance modeling.[67] Addressing reliability for data necessarily involves the inclusion of redundant information, either in the form of duplication of the data to be stored or transmitted, or by generating information that can be used to rebuild lost or damaged data in the case of a fault. The typical way of addressing reliability concerns in large-scale storage systems has been through the generation of syndromes that can be used to detect faults, prevent those faults from manifesting as failures, and repair those failures when new resources are available. The most basic type of syndrome that can be allocated is that of XOR parity.[68] Given a set of data to be stored or transmitted, this data is split into abstract sub-units of fixed or dynamic size called "blocks." We can treat each of these blocks as a vector of bytes and generate a *syndrome* by performing some calculation on each byte in the vector. In order to tolerate the loss of a single block in some set of n blocks we must first compute a syndrome P that allows for the recovery of any lost block within the set. One of the simplest methods for doing so is XOR parity:

$$P = B_0 \oplus B_1 \oplus B_2 \oplus \ldots \oplus B_{n-1}$$

This new block P can then be stored and transmitted with the set, provided each block of the message is transmitted in such a way as to guarantee independence with respect to faults resulting in data loss. The loss of any one block, including the one containing P will not result in the loss of data. If some block B_j fails to transmit, or is corrupted, operations can still be performed on the set as normal by recomputing the value of the lost block. Given a loss of a block not containing P, we rebuild some lost block B_j as

$$B_j = B_0 \oplus B_1 \oplus B_2 \oplus \ldots \oplus B_{j-1} \oplus B_{j+1} \oplus \ldots \oplus B_{n-1} \oplus P$$

(where $2 < j < n-1$ for this example, but without loss of generality for other cases). If P is lost, the data can be used normally, and P can be recomputed as before.

In order to tolerate the loss of any two blocks, two independent syndromes must be calculated, here referred to as P and Q. We utilize the algebra of a Galois field $\mathbf{GF}(2^8)$[69] for the computation of Q to ensure the orthogonality of its construction. We utilize elements g called generators of the Galois field such that g^n doesn't repeat until it has exhausted all elements of the field except {00}, where any numeral in {} is a hexadecimally represented Galois field element. A full discussion of Galois field algebra is fully explored in the literature.[70] For n blocks where $n \leq 255$ we compute:

$$\begin{aligned} P &= B_0 \oplus B_1 \oplus B_2 \oplus \ldots \oplus B_{n-1} \\ Q &= g^0 \cdot B_0 \oplus g^1 \cdot B_1 \oplus g^2 \cdot B_2 \oplus \ldots \oplus g^{n-1} \cdot B_{n-1} \end{aligned}$$

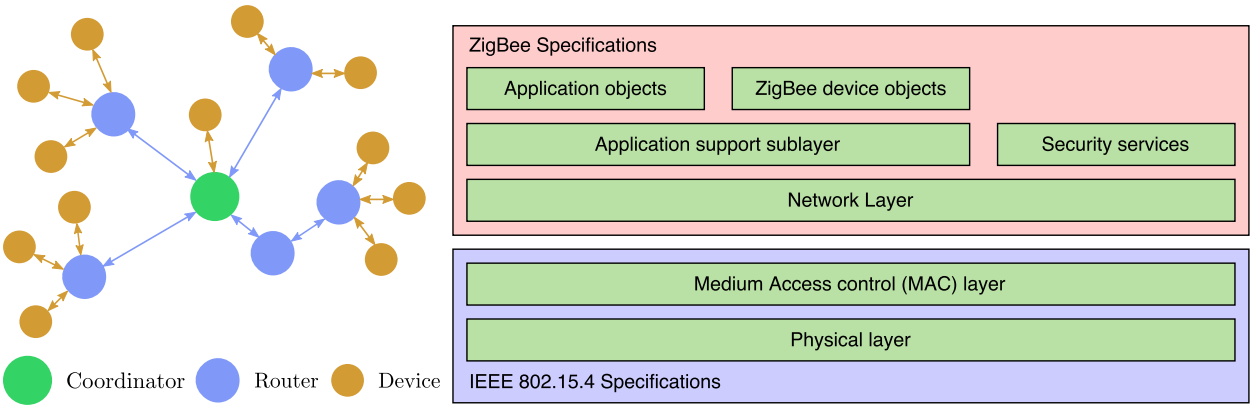
The loss of a single block can be recovered using the normal XOR parity method described previously. The loss of P or Q can be recovered simply by recomputing using the above formulas. The loss of any single non-parity block, and the loss of Q can be recovered by first recovering the non-parity block using XOR parity, and then recomputing Q. Recovering P, or the loss of two data drives is somewhat more involved, and the discussion of the method is left to the literature.[69]

II.F. ZigBee Protocol

ZigBee is an IEEE 802.15.4 based specification protocol for small area networks. It is intended to be simpler and less expensive compared to Bluetooth or Wifi. Its transmission distance is up to 10-100 meters, dependent on power output and environmental characteristics. The transmitted data is secured by 128-bit symmetric encryption keys and has a maximum theoretical data rate of 250 kbits/sec. The ZigBee network layer natively supports star and tree networks, and generic mesh networking. Every ZigBee network must have one coordinator device, tasked with network creation, the control of its parameters and basic maintenance. The network is self-organized. It uses a node called the *coordinator* that acts as the primary ZigBee node and initiates network formation. The sensor nodes, or *end-devices* collect system values and send to the coordinator. The sensor nodes can connect directly to the coordinator or through an intermediate *collector* or *router*. Figure 4a shows communication between different nodes of a ZigBee network using a mesh topology. Coordinators and routers are always-on devices, whereas, end-devices look for an available coordinator when they power up. Figure 4b shows an abstracted version of the ZigBee protocol stack split between the ZigBee specification and the IEEE 802.15.4 specification.

III. Trustworthy Messaging for Aviation Systems

Two primary challenges present themselves when attempting to enable communication between systems over wireless networks in mission- and life-critical aviation systems, privacy and fault-tolerance. Unlike wired systems, wireless systems are prone to spoofing and eavesdropping meaning it is critical that we are able to authenticate messages received, and hide the contents of messages from those without the appropriate privileges. In addition, we must ensure messages can survive natural faults occurring within the system, and attempts by adversaries to induce faults and corrupt messages to prevent transmission. To solve these challenges and build a trustworthy wireless messaging system for next-generation aviation systems we propose a novel, dynamic, messaging structure that combines fault-tolerance using parity, and Galois-field based



(a) (b)

Figure 4: (a) Example ZigBee network using a mesh topology showing communication between different nodes. (b) Layered architecture of the ZigBee 802.15.4 protocol stack

syndromes to improve fault-tolerance, PEKS-like[9,15] regular expression based keywords and meta-keywords for privacy-preserving search[71,72] and a dynamically-sized payload as pictured in Figure 5.

Information to be transmitted is first divided into packets as normal with the ZigBee protocol, but additional packets are inserted to form a set of packets that we refer to as a *burst*. Bursts are evaluated together for fault-tolerance reasons. Each burst of packets consists of a variable number of N payload packets followed by two packets for fault-tolerance purposes computed using XOR- and Galois-field syndrome methods as discussed in Section II.E. The number of payload packets per burst can be configured at run-time and controlled adaptively to trade-off effective bandwidth for fault-tolerance and vice-versa. This gives our system the ability to alter fault-tolerance and throughput capabilities in response to changing conditions on the aircraft. When the network communication path becomes more congested, or in response to potential denial of service attacks, our system can drop the fault-tolerance of individual bursts to ensure packets can be transmitted effectively in the available bandwidth, and to ensure real-time deadlines are met. When bandwidth is plentiful, fault-tolerance can be improved. Figure 6 shows the small modification necessary to enable burst-formation and interpretation with ZigBee Packets.

Members of a burst allocate the first 1 or 2 octets of their packet frame's payload (an overhead of less than 1.5%) to represent the payload index (i.e. the packet's index within a burst) and in the case of the first packet in a payload, the size of payload. Normal data follows. Since this new format adheres to existing ZigBee standards, it can be used with COTS equipment and requires only a software interpreter at the end points of routed messages to interpret. If packets are lost at any point during transmission, they can be rebuilt with high probability either by the ultimate receiver, or by intermediate nodes in the mesh network.

Each data object being transmitted is further modified via encryption using current standards for end-to-end encryption, such as ECCDHE.[73,74] Included with the transmission are trap-door encrypted keywords that can be used by intermediaries to route and index information securely using the RESeED framework for regular-expression based search over encrypted data.[71,72] This allows the data to be transmitted in a semi-oblivious manner. Individual nodes can route messages from any party along pathways without knowing their eventual destination, payload contents, or purpose. Authorized end-users and nodes can filter based on these trap-doors, or query nodes for available data without first decrypting the stored data. This prevents adversaries from determining the content of messages, their origination, and their purpose.

Our proposed modifications to the ZigBee wireless standard provides additional trustworthy capabilities necessary for the specific challenges of next-generation aerospace systems, such as those shown in Figure 7. Given the large physical footprint of most aerospace systems, the ZigBee protocol cannot be implemented in a universal broadcast fashion in which every device can communicate directly with every other device in the large mesh network. Messages will instead have to be relayed, sometimes over multiple hops. It is even possible that every intermediary that relays a message may not have the same permissions, or trust. Individual nodes may be sourced from different contractors, may have been patched for vulnerabilities at different dates, and may be subject to different threats, such as backdoors or physical compromise. In Figure 7 a pitot tube (A) on the wing of the aircraft needs to communicate with the cockpit (C) to report

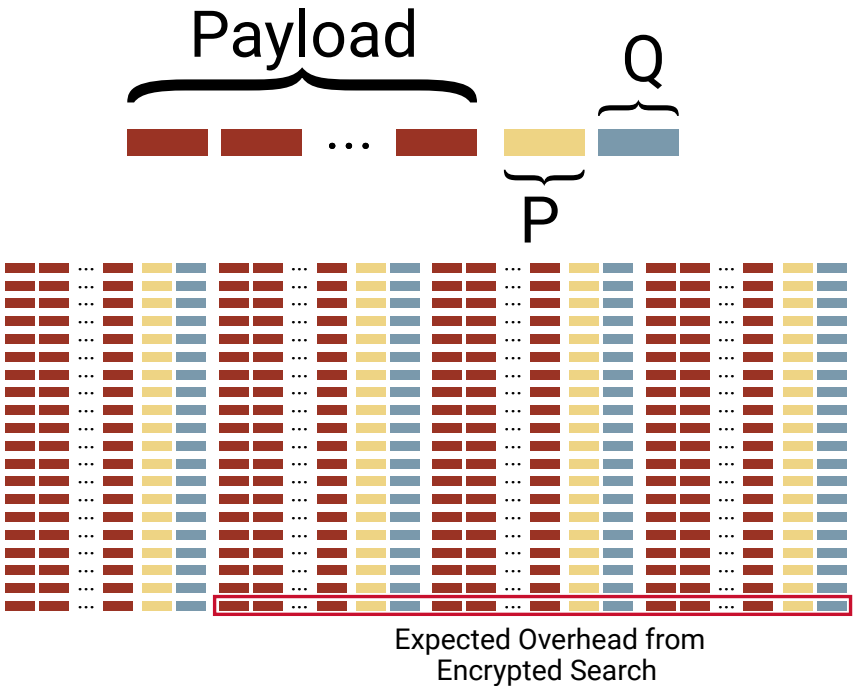


Figure 5: Anatomy of a file transmitted using secure, and reliable, packet bursts. Each burst consists of an adjustable number of payload packets carrying data or meta-data, and parity packets. Extra encrypted search meta-data is included in additional bursts. The example in the diagram represents the expected proportion of extra bursts that must be transmitted to implement searchable encryption mechanisms.

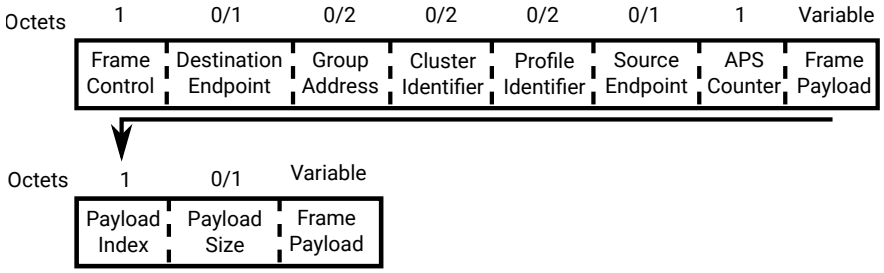


Figure 6: Modified format of ZigBee packet as part of a burst.

observed airspeed. Because the cockpit is out of range for the transmitter (either permanently due to distance, or temporarily due to interference) it must relay its message through an unpatched intermediary (B) that represents another subsystem on the aircraft equipped with a ZigBee transmitter. An adversary (D) is attempting to eavesdrop on packets from (A), or potentially to forge messages from (A) through the compromised node (B). Because the sensor is encrypting all traffic with (C) through public/private key-pair encryption with searchability the adversary is thwarted in his attempts.

The proposed modifications allow for recovery from faults without decrypting the underlying message, and allow for fault recovery either at intermediate nodes, or the eventual end recipient of a message. Because the frame payload of each message is encrypted, adversaries cannot determine the origination of packets, their contents. Onion routing[75] can be employed as part of existing modifications of the ZigBee standard[76] to obscure message origination and destination.[77]

IV. Formal Framework

We present a formal framework for fault analysis in a basic ZigBee network. Network protocols are suitable candidates for contract-based verification since their layered architecture makes them amenable to

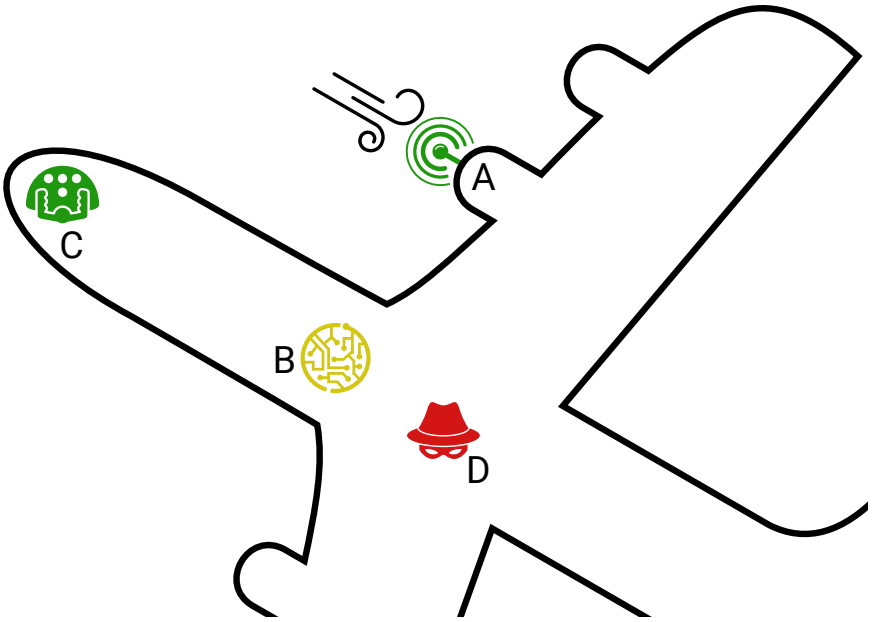


Figure 7: An example scenario where trustworthy messaging is needed. The pitot tube indicated by A wants to transmit information on the current measured air speed to the cockpit and autopilot indicated by C, but lacks the necessary wireless range. It must therefor relay its message through the partially trusted intermediary, B, which is another subsystem of the aircraft. The adversary, D, is attempting to spoof messages from A, eavesdrop on A's messages, and cause messages from A and B to fail to successfully transmit.

compositional modeling. As the first step, we abstract the ZigBee protocol to a form suitable for model checking. The models we use for fault analysis contain each component of the protocol stack of Figure 4b except the *Security Services* sub components. Figure 8 shows the abstracted communication network model for a ZigBee end-device and coordinator.

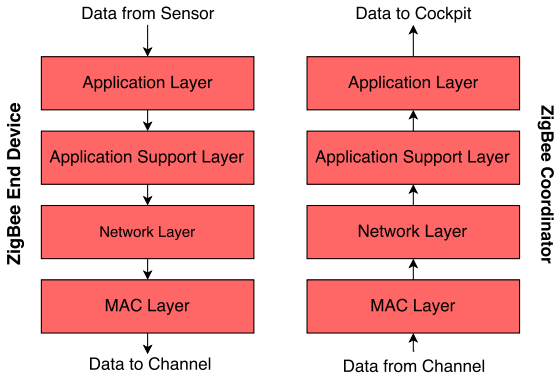


Figure 8: Abstracted and layered ZigBee communication network model showing data flow across layers.

The framework uses three tools: OCRA[53] for component based modeling, contract-based design and refinement, nuXmv[37] for model checking, and xSAP[78] for safety assessment and analysis. Figure 9 shows the nominal framework model of the ZigBee wireless communication system. The framework can be adapted to any wireless network protocol by modifying the behavior of the layered wireless network components in the framework. For wired systems, the framework can be modified by removing the wireless components altogether.

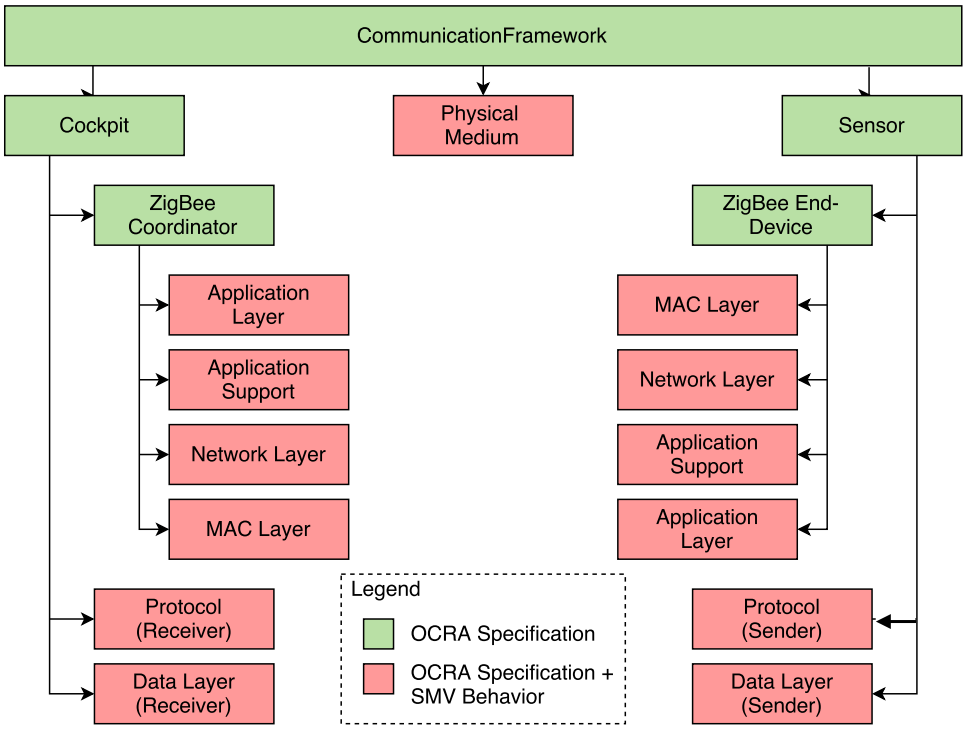


Figure 9: Nominal framework model for the ZigBee wireless communication system

Table 1: Faults associated with wired network analyzed (S: sensor, R: cockpit)

Fault	Description	Mode	Authority
W1	Physical medium breaks	Permanent	Physical Medium
C1	Error recovery mechanism fails	Transient	Protocol (S/R)
S2	Sensor fails	Permanent	Data Layer (S)

IV.A. Nominal Models

For the nominal case, i.e. no faults in the system, the behavior of each component in the framework is simply to output whatever appears at the input. This is verified by using a simple $assume \rightarrow guarantee$ requirement where $assume = input$ and $guarantee = output$. OCRA is used to generate the composite SMV file that connects all the components of the model into one SMV file.

IV.B. Extended Models

For the extended case, the SMV behavior of selected components is modified to include faults and OCRA is run again to generate the composite SMV file. In the current phase of the work, some of the faults are trivial. Others were derived from a subset of faults associated with a ZigBee network covered in existing literature. The framework can be used with any wireless network whose faults are known a priori.

Faults in a wired system In the extended model for the wired system, the only fault we modeled was of a wire breaking (W1) as shown in Table 1. The other faults C1 and S2 are general faults associated with the sensor system. C1 models the failure of the error recovery mechanism, whereas S2 is a harder fault and models the failure of the sensor itself.

Table 2: Faults associated with wired network analyzed in this work (S: sensor, R: cockpit)

Fault	Description	Mode	Authority
Z1	Signal interference	Transient	Physical Medium
Z2	End-Device not discoverable	Transient	Network Layer (S)
Z3	Coordinator cannot accept new connections	Transient	Network Layer (R)
Z4	Coordinator fails to set up network	Permanent	Application Layer (R)
C1	Error recovery mechanism fails	Transient	Protocol (S/R)
S2	Sensor fails	Permanent	Data Layer (S)

FAULTS IN THE ZIGBEE SYSTEM The faults associated with the ZigBee wireless system are a subset of the faults associated real world ZigBee networks. Fault Z1 models signal corruption due to electromagnetic interference in the communication spectrum. Faults Z2, Z3 and Z4 are general faults associated with a ZigBee network and model the end-device not being discoverable, coordinator not accepting new connections, and coordinator failing to set up the network, respectively. Table 2 shows the faults modeled in our extended wireless system.

V. Experimental Results

In order to assess fault tolerance, we model the occurrence of failures predicted by our fault tree constructions with the assumption that packet failures satisfy the Markovian assumptions, i.e. the process of packet failures is stateless and independent.[79] Under this assumption we evaluate our failures as a Poisson process[80] and solve numerically. Our primary interest is in evaluating the probability that a burst suffers 3 or more failures, as such situations are unrecoverable under our current error correction and fault-tolerance. This is primarily a function of the burst payload size, and our expectations are that as the number of payload packets increase per burst, fault-tolerance will drop, but bandwidth utilization will improve.

The probability that a burst will fail is given by

$$P(\text{fail}_{\text{burst}}) = 1 - \frac{\Gamma(\lfloor k+1 \rfloor, \lambda)}{\lfloor k \rfloor !}$$

where $\Gamma()$ is the incomplete-gamma function,[81] k is the maximum number of faults we can tolerate without burst failure, and λ is the expected number of failures per burst. For our purposes we work with $k = 2$ due to the limitations of our syndrome mechanisms. We find λ as a function of the data-transmission rate r. This rate, r is either 20kbps or 250kbps, depending on the band being used. We make the assumption that r is 20kbps for the results presented in this paper, but also computed the results for 250kbps with similar results. The per packet failure rate is given in this equation as μ. The configurable number of packets per burst is represented as b. Lastly the size of the packet is given as s and assumed to be 512 bits for our evaluation. With these substitutions our final expression for λ as

$$\lambda = \frac{\mu \cdot r}{s} \cdot \frac{b \cdot s}{r} = \mu b.$$

V.A. Formal Fault Analysis

In order to generate estimates for the per packet failure rate μ, both for our experiment, and as a tool for systems to deploy for runtime estimation during operation, we designed models for the nominal and off-nominal behavior of our ZigBee system. The nominal system was organized using OCRA, which then generates a composite SMV file. Faults were injected manually in the composite nominal SMV file using

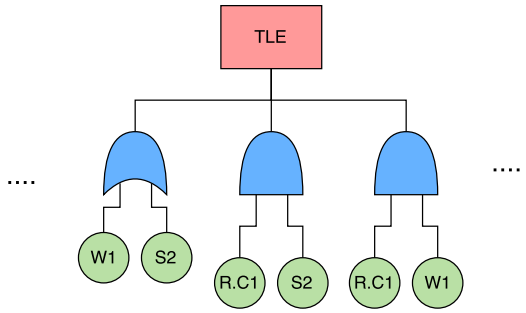
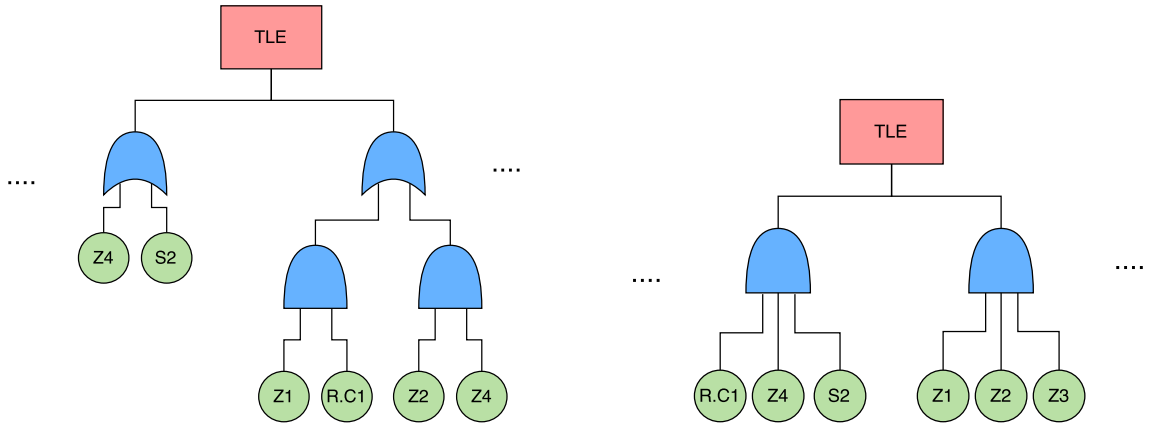


Figure 10: Fault tree using minimal cut sets for the extended wired system.

$$
\begin{aligned}
CutSets &= (\{W1, S2, R.C1, S2, R.C1, W1\}, \{W1, R.C1, S2, R.C1, W1\}, \\
&\quad \{S2, R.C1, S2, R.C1, W1\}, W1, S2, \{R.C1, S2\}, \{R.C1, W1\} \ldots) \\
MinCutSet &= (W1, S2, \{R.C1, S2\}, \{R.C1, W1\})
\end{aligned}
$$

Figure 11: Cut Sets and Minimal Cut Sets for fault tree in Figure 10.



(a) Sample fault tree 1

(b) Sample fault tree 2

Figure 12: Fault tree using minimal cut sets for the extended wired system.

the fault libraries of xSAP. All faults were modeled as simple Boolean variables that are *true* when a fault event occurs, and are *false* otherwise. All the faults modeled are non-deterministic in nature. The top-level requirement for our nominal and extended system is "Data transmitted from the cockpit is received by the sensor". In LTL, the requirement is specified as:

$$
\begin{aligned}
&\textsc{Always}(Sensor.DataLayer.input \rightarrow \quad \textsc{Eventually}(Cockpit.DataLayer.output \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad (Cockpit.DataLayer.output = Sensor.DataLayer.input)))
\end{aligned}
$$

For fault tree analysis, our top-level event was the negation of our top-level requirement. Figure 10 and Figure 11 show a sample fault tree and the cut set and minimal cut sets generated by xSAP for the extended wired model.

For the ZigBee wireless network, Figure 12 shows fault trees for the extended model, and Figure 13 and Figure 14 show the cut set and minimal cut sets generated by xSAP. After the points of failure are determined, a failure function assigns probabilities to individual faults. The overall failure probability needs to be equal to or less than the failure probability of the wired system. Similarly, multiple wireless systems can also be compared.

$$
\begin{aligned}
CutSets &= (\{Z4, S2, Z1, R.C1, Z2\}, \{Z4, Z1, R.C1, Z2\}, \\
&\quad \{S2, Z1, R.C1, Z2\}, Z4, S2, \{Z1, R.C1\}, \{Z2, Z4\} \ldots) \\
MinCutSet &= (Z4, S2, \{Z1, R.C1\}, \{Z2, Z4\})
\end{aligned}
$$

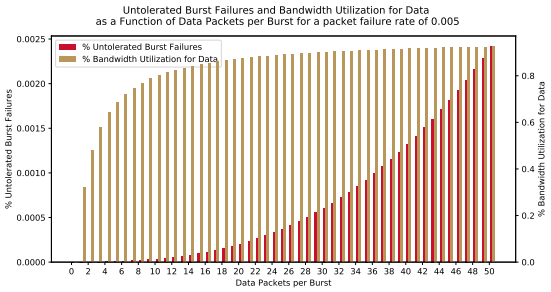
Figure 13: Cut Sets and Minimal Cut Sets for fault tree in Figure 12a.

$$
\begin{aligned}
CutSets &= (\{R.C1, Z4, S2, Z1, Z2, Z3\}, \{R.C1, Z4, S2\}, \\
&\quad \{Z1, Z2, Z3\} \ldots) \\
MinCutSet &= (\{R.C1, Z4, S2\}, \{Z1, Z2, Z3\})
\end{aligned}
$$

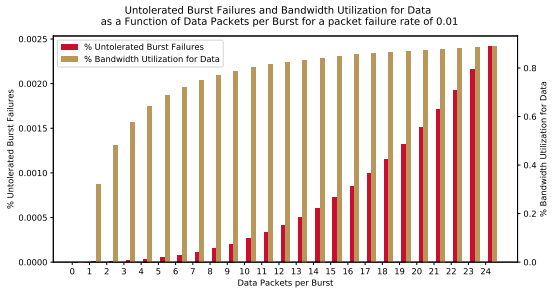
Figure 14: Cut Sets and Minimal Cut Sets for fault tree in Figure 12b.

V.B. Results

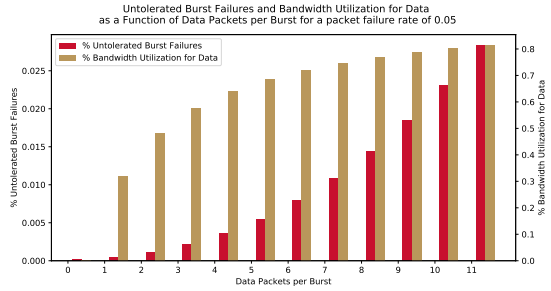
The empirical results of our analysis are given in Figures 15a to 15d. Using our numerical model of fault tolerance we plotted the expected percentage of untolerated burst failures occurring over any transmission epoch, and the percentage of bandwidth utilized for data transmission, accounting for the 4% overhead for RESeED, our parity packets included in each burst, and the additional octets needed to encode the burst in the existing ZigBee Standard.



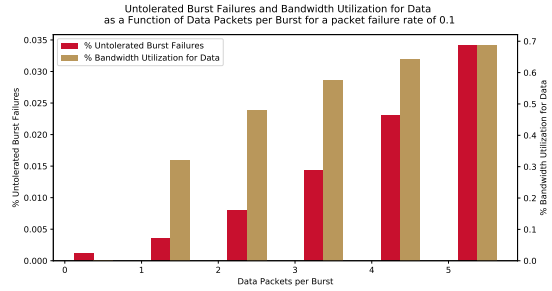
(a) Per Packet Failure Rate $\mu = 0.005$



(b) Per Packet Failure Rate $\mu = 0.01$



(c) Per Packet Failure Rate $\mu = 0.05$



(d) Per Packet Failure Rate $\mu = 0.1$

Figure 15: A comparison of the impact on differing numbers of data packets per burst on the percentage of bursts with untolerated failures, and the percentage of bandwidth utilized for productive data transfer for per packet failure rates of $\mu \in \{0.005, 0.01, 0.05, 0.1\}$

We evaluated our system for several different configurations in terms of payload packets per burst, and for four different failure rates of ZigBee packets. The literature[5] suggests that failure rates can vary from negligible rates (as in Figures 15a and 15b) to as high as 10% in extreme circumstances (Figure 15d). We present results where the failure rate per burst is no higher than roughly 0.03, representing an expected loss of at most 3% of bursts during transmission to show the impact on our tunable parameter (data packets per

burst) on bandwidth utilization for data, and our ability to tolerate packet failures. As can be seen from the plots, our system is not only robust in the presence of failures, but the tunability allows systems to optimize performance with respect to the most important metric for a given situation, regardless of the current failure rate experienced by packets.

VI. Conclusions and Future Work

We demonstrate a proof of concept for the development of a framework for a comparative fault analysis of wired and wireless communication networks. The work is still incomplete in terms of quantitative evaluation. The formal framework model is plug-and-play in the sense that new wireless protocol behavior models can be plugged in easily in to the composite system model with minor modifications.

Future extensions of the work include better understanding how xSAP calculates minimal cut sets, quantitative assessment failure probabilities, add more behavior and fault extensions to the models and removing some obvious extensions. Moreover, for wired networks, performing Common Cause Analysis (CCA) might lead to useful results since wires are prone to damage due to surroundings. A thorough analysis of the fault tolerant architectures by adding extensions for redundant end devices (sensor), coordinators (cockpit device) and communication medium (wires, different frequencies) will help designers in identifying optimum fault tolerance techniques. Extending the framework to include more wireless communication protocols will help better identify wired components of the aircraft than can be migrated to wireless. A desirable extension will be automatic introduction of fault tolerant architectures to achieve a desired probability.

We intend to implement our proposed extension to the ZigBee framework for dynamic fault-tolerance and bandwidth utilization to show its efficacy in practice, COTS deployability, and to demonstrate how simple automated reasoning on-board an aircraft could be used to estimate the current packet failure rate and adaptively modify the burst configuration to meet mission-critical goals for transmission.

References

[1]Canaday, H., "War on Wiring," *Aerospace America*, May 2017, pp. 24–27.

[2]Attia, S. B., Cunha, A., Koubâa, A., and Alves, M., "Fault-tolerance mechanisms for ZigBee wireless sensor networks," *Work-in-Progress (WiP) session of the 19th Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa, Italy*, No. 1, 2007, pp. 37–40.

[3]Wan, J., Wu, J., and Xu, X., "A novel fault detection and recovery mechanism for zigbee sensor networks," *Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on*, Vol. 1, IEEE, 2008, pp. 270–274.

[4]Alena, R., Ellis, S. R., Hieronymus, J., and Maclise, D., "Wireless Avionics and Human Interfaces for Inflatable Spacecraft," *Aerospace Conference, 2008 IEEE*, IEEE, 2008, pp. 1–16.

[5]Alena, R., Gilstrap, R., Baldwin, J., Stone, T., and Wilson, P., "Fault tolerance in ZigBee wireless sensor networks," *Aerospace Conference, 2011 IEEE*, IEEE, 2011, pp. 1–15.

[6]Song, D. X., Wagner, D., and Perrig, A., "Practical techniques for searches on encrypted data," *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, IEEE, 2000, pp. 44–55.

[7]Goh, E.-J., "Secure Indexes," Cryptology ePrint Archive, 2003, Report 2003/216.

[8]Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R., "Searchable symmetric encryption: improved definitions and efficient constructions," *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, 2006, pp. 79–88.

[9]Boneh, D. and Waters, B., "Conjunctive, subset, and range queries on encrypted data," *Theory of cryptography*, Springer, 2007, pp. 535–554.

[10]Wang, C., Ren, K., Yu, S., and Urs, K. M. R., "Achieving usable and privacy-assured similarity search over outsourced cloud data," *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM '12, 2012, pp. 451–459.

[11]Ibrahim, A., Jin, H., Yassin, A. A., Zou, D., and Xu, P., "Towards Efficient Yet Privacy-Preserving Approximate Search in Cloud Computing," *The Computer Journal*, Vol. 56, No. 5, May 2013, pp. 1–14.

[12]Hjaltason, G. R. and Samet, H., "Properties of Embedding Methods for Similarity Searching in Metric Spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, May 2003, pp. 530–549.

[13]Kamara, S. and Lauter, K., "Cryptographic cloud storage," *Financial Cryptography and Data Security*, Springer, 2010, pp. 136–149.

[14]Hofheinz, D. and Weinreb, E., "Searchable encryption with decryption in the standard model." *IACR Cryptology ePrint Archive*, Vol. 2008, 2008, pp. 423.

[15]Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G., "Public key encryption with keyword search," *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2004, pp. 506–522.

[16]Boneh, D. and Franklin, M., "Identity-based encryption from the Weil pairing," *Advances in CryptologyâĂŤCRYPTO 2001*, Springer, 2001, pp. 213–229.

[17]Zhao, Y. and Rozier, K. Y., "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System," *Proceedings of the 12th International Workshop on Automated Verification of Critical Systems (AVoCS 2012)*, Vol. 53 of *Electronic Communications of the EASST*, European Association of Software Science and Technology, 2012.

[18]Zhao, Y. and Rozier, K. Y., "Formal Specification and Verification of a Coordination Protocol for an Automated Air Traffic Control System," *Science of Computer Programming Journal*, Vol. 96, No. 3, December 2014, pp. 337–353.

[19]Zhao, Y. and Rozier, K. Y., "Probabilistic Model Checking for Comparative Analysis of Automated Air Traffic Control Systems," *Proceedings of the 33rd IEEE/ACM International Conference On Computer-Aided Design (ICCAD 2014)*, IEEE/ACM, San Jose, California, U.S.A., November 2014, pp. 690–695.

[20]Mattarei, C., Cimatti, A., Gario, M., Tonetta, S., and Rozier, K. Y., "Comparing Different Functional Allocations in Automated Air Traffic Control Design," *Proceedings of Formal Methods in Computer-Aided Design (FMCAD 2015)*, IEEE/ACM, Austin, Texas, U.S.A, September 2015.

[21]Gario, M., Cimatti, A., Mattarei, C., Tonetta, S., and Rozier, K. Y., "Model Checking at Scale: Automated Air Traffic Control Design Space Exploration," *Proceedings of 28th International Conference on Computer Aided Verification (CAV 2016)*, Vol. 9780 of *LNCS*, Springer, Toronto, ON, Canada, July 2016, pp. 3–22.

[22]Groce, A., Havelund, K., Holzmann, G., Joshi, R., and Xu, R.-G., "Establishing flight software reliability: Testing, model checking, constraint-solving, monitoring and learning," 2014.

[23]Mehlitz, P. C., "Trust your model-verifying aerospace system models with Java pathfinder," *Aerospace Conference, 2008 IEEE*, IEEE, 2008, pp. 1–11.

[24]Can, A. B., Bultan, T., Lindvall, M., Lux, B., and Topp, S., "Eliminating synchronization faults in air traffic control software via design for verification with concurrency controllers," *Automated Software Engineering*, Vol. 14, No. 2, 2007, pp. 129–178.

[25]Munoz, C., Carreño, V., and Dowek, G., "Formal analysis of the operational concept for the small aircraft transportation system," *Rigorous Development of Complex Fault-Tolerant Systems*, Springer, 2006, pp. 306–325.

[26]Bozzano, M., Cimatti, A., Katoen, J.-P., Nguyen, V. Y., Noll, T., and Roveri, M., "The COMPASS approach: Correctness, modelling and performability of aerospace systems," *Computer Safety, Reliability, and Security*, Springer, 2009, pp. 173–186.

[27]Chan, W., Anderson, R. J., Beame, P., Burns, S., Modugno, F., Notkin, D., and Reese, J. D., "Model checking large software specifications," *Software Engineering, IEEE Transactions on*, Vol. 24, No. 7, 1998, pp. 498–520.

[28]Sreemani, T. and Atlee, J. M., "Feasibility of model checking software requirements: A case study," *Computer Assurance, 1996. COMPASS'96, Systems Integrity. Software Safety. Process Security. Proceedings of the Eleventh Annual Conference on*, IEEE, 1996, pp. 77–88.

[29]von Essen, C. and Giannakopoulou, D., "Analyzing the next generation airborne collision avoidance system," *Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2014, pp. 620–635.

[30]for Aeronautics (RTCA), R. T. C., "DO-178B: Software Considerations in Airborne Systems and Equipment Certification," December 1992.

[31]for Aeronautics (RTCA), R. T. C., "DO-254: Design Assurance Guidance for Airborne Electronic Hardware," April 2000.

[32]Clarke, E., Emerson, E., and Sistla, A., "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications," *ACM TOPLAS*, Vol. 8, No. 2, Jan 1986, pp. 244–263.

[33]Queille, J. and Sifakis, J., "Specification and Verification of Concurrent Systems in Cesar," *Proc. 5th Int'l Symp. on Programming*, Vol. 137 of *LNCS*, Springer, 1982, pp. 337–351.

[34]Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., and Tonetta, S., "The nuXmv Symbolic Model Checker," *CAV*, 2014, pp. 334–342.

[35]Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M., "NuSMV: A New Symbolic Model Checker," *International Journal of Software Tools for Technology Transfer (STTT)*, Vol. 2, No. 4, 2000, pp. 410–425.

[36]R. Cavada, A. C., Jochim, C., Keighren, G., Olivetti, E., Pistore, M., Roveri, M., and Tchaltsev, A., "NuSMV 2.4 User Manual," Tech. rep., CMU/ITC-irst, 2005.

[37]Bozzano, M., Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., and Tonetta, S., "nuXmv 1.0 User Manual," Tech. rep., FBK - Via Sommarive 18, 38055 Povo (Trento) âĂŞ Italy, 2014.

[38]Raimondi, F., Lomuscio, A., and Sergot, M. J., "Towards Model Checking Interpreted Systems," *FAABSÂć02, LNAI 2699*, Springer, 2002, pp. 115–125.

[39]Gribaudo, M., Horvath, A., Bobbio, A., Tronci, E., Ciancamerla, E., and Minichino, M., "Model-checking Based on Fluid Petri Nets for the Temperature Control System of the ICARO Co-generative Plant," Tech. rep., SAFECOMP, 2434, LNCS, 2002.

[40]Tribble, A. and Miller, S., "Software Safety Analysis of a Flight Management System Vertical Navigation Function-A Status Report," *DASC*, 2003, pp. 1.B.1–1.1–9 v1.

[41]Choi, Y. and Heimdahl, M., "Model Checking Software Requirement Specifications Using Domain Reduction Abstraction," *IEEE ASE*, 2003, pp. 314–317.

[42]Miller, S. P., Tribble, A. C., Whalen, M. W., Per, M., and Heimdahl, E., "Proving the Shalls," *STTT*, Vol. 8, No. 4-5, 2006, pp. 303–319.

[43]Miller, S., "Will This Be Formal?" *TPHOLs 5170*, Springer, 2008, pp. 6–11.

[44]Yoo, J., Jee, E., and Cha, S., "Formal Modeling and Verification of Safety-Critical Software," *Software, IEEE*, Vol. 26, No. 3, 2009, pp. 42–49.

[45]Gan, X., Dubrovin, J., and Heljanko, K., "A Symbolic Model Checking Approach to Verifying Satellite Onboard Software," *Science of Computer Programming (2013)*, March 2013.

[46]Lahtinen, J., Valkonen, J., Björkman, K., Frits, J., Niemelä, I., and Heljanko, K., "Model checking of safety-critical software in the nuclear engineering domain," *Reliability Engineering & System Safety*, Vol. 105, No. 0, 2012, pp. 104 – 113.

[47]Bozzano, M., Cimatti, A., Pires, A. F., Jones, D., Kimberly, G., Petri, T., Robinson, R., and Tonetta, S., "Formal Design and Safety Analysis of AIR6110 Wheel Brake System," *Proceedings of the 27th International Conference on Computer Aided Verification (CAV)*, Springer, July 2015.

[48]McMillan, K., "The SMV Language," Tech. rep., Cadence Berkeley Lab, 1999.

[49]de Moura, L., Owre, S., Rueß, H., Rushby, J., Shankar, N., Sorea, M., and Tiwari, A., "SAL 2," *CAV*, edited by R. Alur and D. Peled, Vol. 3114 of *LNCS*, Springer, Boston, MA, Jul 2004, pp. 496–500.

[50]Brayton, R., Hachtel, G., Sangiovanni-Vincentelli, A., Somenzi, F., Aziz, A., Cheng, S.-T., Edwards, S., Khatri, S., Kukimoto, T., Pardo, A., Qadeer, S., Ranjan, R., Sarwary, S., Shiple, T., Swamy, G., and Villa, T., "VIS: a system for verification and synthesis," *CAV, Proc. 8th Int'l Conf*, Vol. 1102 of *LNCS*, Springer, 1996, pp. 428–432.

[51]Rozier, K., "Linear Temporal Logic Symbolic Model Checking," *Computer Science Review Journal*, Vol. 5, No. 2, May 2011, pp. 163–203.

[52]Cimatti, A. and Tonetta, S., "A property-based proof system for contract-based design," *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, IEEE, 2012, pp. 21–28.

[53]Alessandro Cimatti, Michele Dorigatti, S. T., "OCRA: Othello Contracts Refinement Analysis, Version 1.3," *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, IEEE, 2013, pp. 702–705.

[54]Cimatti, A. and Tonetta, S., "Contracts-refinement proof system for component-based embedded systems," *Science of Computer Programming*, Vol. 97, 2015, pp. 333–348.

[55]Gómez-Martínez, E., Rodríguez, R. J., Elorza, L. E., Rezabal, M. I., and Earle, C. B., "Model-based verification of safety contracts," *International Conference on Software Engineering and Formal Methods*, Springer, 2014, pp. 101–115.

[56]Cimatti, A. and Tonetta, S., "A Temporal Logics Approach to Contract-Based Design," *2016 Architecture-Centric Virtual Integration (ACVI)*, April 2016, pp. 1–3.

[57]Quinton, S. and Graf, S., "Contract-based verification of hierarchical systems of components," *Software Engineering and Formal Methods, 2008. SEFM'08. Sixth IEEE International Conference on*, IEEE, 2008, pp. 377–381.

[58]Graf, S., Passerone, R., and Quinton, S., "Contract-based reasoning for component systems with rich interactions," *Embedded Systems Development*, Springer, 2014, pp. 139–154.

[59]Nuzzo, P., Sangiovanni-Vincentelli, A. L., Bresolin, D., Geretti, L., and Villa, T., "A Platform-Based Design Methodology With Contracts and Related Tools for the Design of Cyber-Physical Systems," *Proceedings of the IEEE*, Vol. 103, No. 11, Nov 2015, pp. 2104–2132.

[60]Schroeder, B. and Gibson, G. A., "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you," *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)*, 2007, pp. 1–16.

[61]Rozier, E., Belluomini, W., Deenadhayalan, V., Hafner, J., Rao, K., and Zhou, P., "Evaluating the impact of undetected disk errors in raid systems," *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, IEEE, 2009, pp. 83–92.

[62]Hafner, J. L., Deenadhayalan, V., Belluomini, W., and Rao, K., "Undetected disk errors in RAID arrays," *IBM Journal of Research and Development*, Vol. 52, No. 4.5, 2008, pp. 413–425.

[63]Wallace, G., Douglis, F., Qian, H., Shilane, P., Smaldone, S., Chamness, M., and Hsu, W., "Characteristics of backup workloads in production systems," *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FASTâĂŹ12)*, 2012.

[64]Tarasov, V., Kumar, S., Ma, J., Hildebrand, D., Povzner, A., Kuenning, G., and Zadok, E., "Extracting flexible, replayable models from large block traces," *FAST12*, 2012.

[65]Anderson, E., "Capture, conversion, and analysis of an intense NFS workload," *Proccedings of the 7th conference on File and storage technologies*, USENIX Association, 2009, pp. 139–152.

[66]Madhyastha, H. V., McCullough, J. C., Porter, G., Kapoor, R., Savage, S., Snoeren, A. C., and Vahdat, A., "scc: cluster storage provisioning informed by application characteristics and SLAs," *Proceedings of the 10th USENIX conference on File and Storage Technologies*, USENIX Association, 2012, pp. 23–23.

[67]Soundararajan, G., Lupei, D., Ghanbari, S., Popescu, A. D., Chen, J., and Amza, C., "Dynamic resource allocation for database servers running on virtual storage," *Proccedings of the 7th conference on File and storage technologies*, USENIX Association, 2009, pp. 71–84.

[68]Chen, P. M., Lee, E. K., Gibson, G. A., Katz, R. H., and Patterson, D. A., "RAID: High-performance, reliable secondary storage," *ACM Computing Surveys (CSUR)*, Vol. 26, No. 2, 1994, pp. 145–185.

[69]Anvin, H. P., "The mathematics of RAID-6," 2007.

[70]Jacobson, N., *Lectures in Abstract Algebra: III. Theory of Fields and Galois Theory*, Vol. 32, Springer Science & Business Media, 2012.

[71]Salehi, M. A., Caldwell, T., Fernandez, A., Mickiewicz, E., Rozier, E. W., Zonouz, S., and Redberg, D., "Reseed: Regular expression search over encrypted data in the cloud," *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, IEEE, 2014, pp. 673–680.

[72]Salehi, M. A., Caldwell, T., Fernandez, A., Mickiewicz, E., Rozier, E. W., Zonouz, S., and Redberg, D., "RESeED: A secure regular-expression search tool for storage clouds," *Software: Practice and Experience*, 2017.

[73]Joux, A., "A one round protocol for tripartite Diffie–Hellman," *International Algorithmic Number Theory Symposium*, Springer, 2000, pp. 385–393.

[74]Bernstein, D. J., "Curve25519: new Diffie-Hellman speed records," *International Workshop on Public Key Cryptography*, Springer, 2006, pp. 207–228.

[75]Goldschlag, D., Reed, M., and Syverson, P., "Onion routing," *Communications of the ACM*, Vol. 42, No. 2, 1999, pp. 39–41.

[76]Nezhad, A. A., Miri, A., and Makrakis, D., "Location privacy and anonymity preserving routing for wireless sensor networks," *Computer Networks*, Vol. 52, No. 18, 2008, pp. 3433–3452.

[77]Chakrabarty, S., John, M., and Engels, D. W., "Black routing and node obscuring in IoT," *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, IEEE, 2016, pp. 323–328.

[78]Bittner, B., Bozzano, M., Cavada, R., Cimatti, A., Gario, M., Griggio, A., Mattarei, C., Micheli, A., and Zampedri, G., "The xSAP Safety Analysis Platform," *Proceedings of TACAS 2016*, 2016.

[79]Markov, A. A., "The theory of algorithms," *Am. Math. Soc. Transl.*, Vol. 15, 1960, pp. 1–14.

[80]Law, A. M. and Kelton, W. D., *Simulation modeling and analysis*, Vol. 2, McGraw-Hill New York, 1991.

[81]Abramowitz, M. and Stegun, I. A., "Handbook of mathematical functions," *Applied mathematics series*, Vol. 55, No. 62, 1966, pp. 39.