computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

DEPENDABLE INTERNET OF THINGS FOR NETWORKED CARS

Bernhard Großwindhager, Astrid Rupp, Martin Tappler, Markus Tranninger, Samuel Weiser, Bernhard K. Aichernig, Carlo Alberto Boano, Martin Horn, Gernot Kubin, Stefan Mangard, Martin Steinberger, Kay Römer

LEAD Project Dependable Internet of Things, TU Graz, Inffeldgasse 16, 8010 Graz, Austria
roemer@tugraz.at

Abstract: The Internet of Things (IoT) extends the Internet to include also wireless embedded computers that are often equipped with sensors and actuators to monitor and control their physical environment. The IoT is increasingly used for safety-critical applications such as smart factories or networked cars, where a failure of the IoT may lead to catastrophic consequences. The IoT is therefore in urgent need of *dependability*, where reliability, availability, and security properties can be guaranteed even in harsh environments (e.g., radio interference) and under deliberate attacks (e.g., exploiting side channels). In this paper we give an overview of recent research activities in the LEAD project "Dependable Internet of Things in Adverse Environments" towards a dependable IoT, specifically dependable wireless communication and localization using Ultra-Wide-Band technology, secure execution of real-time software, protocol testing and verification, and dependable networked control. We also present the TruckLab testbed, where our research results can be integrated and validated in a platooning use case. In this testbed, model trucks are automatically controlled to follow a lead truck.

Keywords: Dependability, Internet of Things, Car2X, UWB, Security, Testing, Networked Control.

1. INTRODUCTION

The Internet of Things extends the Internet to include also wireless embedded computers that are often equipped with sensors and actuators to monitor and control their physical environment. The IoT is increasingly used also for safety-critical applications such as networked cars or smart factories, where a failure of the IoT may lead to catastrophic consequences. Due to harsh environments, deliberate attacks, and the inherent scale and complexity of the IoT, it is actually likely that such failures will occur. Therefore, the LEAD project "Dependable Internet of Things in Adverse Environments"[1] at Graz University of Technology in Austria investigates all relevant aspects in order to make the IoT *dependable*. The research questions are derived from and validated by a platooning use case, where multiple trucks automatically follow a leader truck. For this, the distance among the trucks has to be accurately estimated, reliable communication among the trucks is needed, the protocols used for communication need to be verified, real-time software needs to be securely executed, and dependable networked control of the driving parameters needs to be achieved.

Towards these goals, this paper gives an overview of recent research activities in the project towards dependable wireless communication and location estimation using ultra-wideband (Sect. 2), secure real-time computing (Sect. 3), automated protocol testing and verification (Sect. 4), as well as networked control (Sect. 5). We also present the TruckLab testbed, which we use to integrate and validate the individual research results.

2. DEPENDABLE UWB COMMUNICATIONS

2.1 POTENTIAL OF UWB TECHNOLOGY

An integral part of a cleaner, safer, and more efficient transportation is Car-2-X technology. It enables vehicles to exchange information wirelessly between each other and the infrastructure. Potential Car-2-X wireless technologies such as IEEE 802.11p are inherently narrowband and therefore highly susceptible to multipath fading and multi-user interference.

Shifting towards ultra-wideband (UWB) has the potential to tackle these limitations. In contrast to

[1] http://dependablethings.tugraz.at

narrowband transceivers, UWB radios spread the signal over a much wider bandwidth (≥500MHz). This results in (i) a higher immunity to multipath fading, (ii) reduced interference, (iii) a high data rate, as well as (iv) superior time-domain resolution allowing for accurate ranging. The latter enables to use the UWB transceiver not only to exchange information, but also to estimate the distance between trucks.

In the cooperative platooning use case presented in Sect. 5, a reliable communication link has to be provided in all scenarios independently of the surrounding environment. Towards this goal, we have developed a novel scheme to adapt UWB physical layer (PHY) settings at runtime in order to provide both reliable communication performance and accurate distance estimation. First, we give an overview of the adaption algorithm (Sect. 2.2). The latter requires a clear understanding of the performance of the PHY settings and which ones to privilege (Sect. 2.3). As the derived ranking is highly dependent on the surrounding environmental conditions, we further characterize the environment and link quality at runtime by using channel impulse response information (Sect. 2.4). Finally, we conclude with an evaluation of the proposed algorithm (Sect. 2.5).

2.2 ADAPTING PHY SETTINGS AT RUNTIME

To date, UWB deployments are using static physical layer settings that remain unchanged over time [1]. Therefore, the systems are unable to cope with changes in the environment, which limits the dependability of communication links. We present an adaptive algorithm that determines at runtime the optimal PHY settings (i.e., the one maximizing reliability while minimizing energy consumption) to be used for dependable communication. Fig. 1 shows the block diagram of the devised scheme.

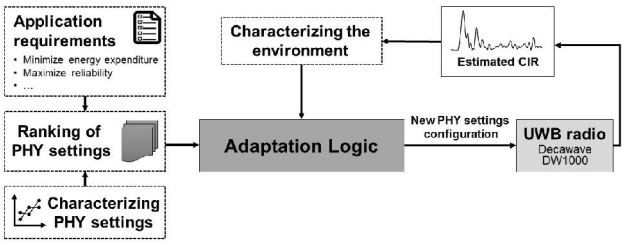


Fig. 1 – Block diagram of the proposed adaptation algorithm.

Based on the application requirements and a characterization of the PHY settings that is presented in Sect. 2.3, a ranking of the settings is determined. The latter serves as an input to the adaptation logic. The second input is the characterization of the environment. In Section 2.4 we use the channel impulse response provided by the UWB radio to detect at runtime non-line-of-sight conditions, the presence of destructive interference, and the received signal strength. Based on this information, the state of the link is derived and, in case of a degrading link, the adaptation logic is triggered. The appropriate PHY configuration is then loaded and shared with neighboring nodes to maintain a highly reliable communication link.

2.3 CHARACTERIZING UWB PHY SETTINGS

Compared to other IoT technologies, UWB transceivers provide not only a higher bandwidth, but also several configurable PHY parameters. The latter drastically affect the reliability and energy consumption of UWB radios. In total, seven different tuning knobs are provided at the physical layer; among others, data rate, carrier frequency, and preamble length. Unfortunately, an extensive investigation of the settings is still missing in the research community. Hence, we have first characterized the performance of all seven UWB PHY parameters experimentally. As an example, Fig. 2 depicts the impact of data rate on packet reception rate (PRR). The x-axis shows the attenuation level of a tunable attenuator that we used to simulate a degrading link. The figure shows that decreasing the data rate from the maximum of 6.8Mbps to 110kbps increases the sensitivity of the receiver by 5.5 dB. Thus, a receiver needs a 5.5 dB weaker signal to sustain the same level of robustness by simply decreasing the data rate. The increased robustness comes at the cost of a longer duration of the payload (64x) and hence a higher energy consumption. The adaptation algorithm introduced in Sect. 2.2 automatically finds a good tradeoff between high reliability and low energy consumption, hence optimizing the dependability of an UWB link.

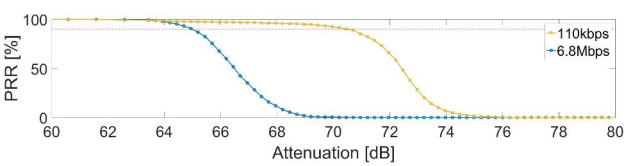


Fig. 2 – Packet reception rate for different attenuation levels as a function of data rate.

2.4 CHARACTERIZING THE ENVIRONMENT

UWB technology provides inherently a more robust communication than narrowband

technologies. Still, a dynamic environment (e.g., the sporadic presence of destructive interference or obstacles blocking the line-of-sight (LOS) between nodes), may highly affect the communication performance. For this purpose, we make use of the channel impulse response (CIR) derived by UWB transceivers to detect such environmental changes and react accordingly. The CIR provides information about the multipath propagation, i.e., radio signals arriving at the receiving antenna via different paths due to reflections from walls and other objects.

The absence of a clear LOS between two communicating devices or vehicles (for example at crossroads or at hilly roads) severely affects the quality of a link due to the reduced received signal strength. Furthermore, in time-of-flight distance measurements, the wrong classification of a non-line-of-sight (NLOS) component as the LOS component leads to positively biased range estimations. Detecting the presence of NLOS condition at run-time by employing the CIR allows to correct these biased estimates [2]. Although UWB transceivers are less affected by multipath fading due to the higher bandwidth, at longer distance the LOS component may still overlap with multipath reflections. This can cause destructive interference and hence heavily degrade a communication link. Detecting the presence of destructive interference is also possible by analyzing the CIR. Fig. 3 shows two CIRs derived from the low-cost UWB transceiver Decawave DW1000 [3]. One was acquired from a highly reliable link (blue line) and the other one inside a deep fade due to destructive interference (green dashed line). In the case of destructive interference, the amplitude of the LOS component (first strong peak) drops significantly. However, the amplitude of the multipath components does not decrease as much as the LOS component, as they do not suffer from destructive interference. Therefore, keeping track of the ratio between the amplitude of the LOS component and of multipath components provides an efficient mechanism to detect destructive interference in an UWB communication link.

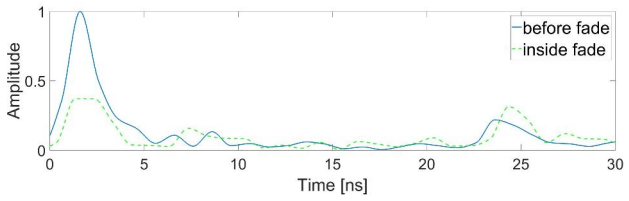


Fig. 3 – Channel impulse response obtained from a Decawave DW1000 UWB transceiver.

Besides the detection of LOS/NLOS condition and destructive interference, we also derive an estimation of the received signal strength from the CIR as an additional indicator of the link quality. This is required as a trigger for the adaptation algorithm presented in Sect. 2.2.

2.5 PERFORMANCE OF THE ADAPTATION LOGIC

We extensively evaluated the performance of the proposed algorithm in highly dynamic environments (fluctuating signal strength) and in the case of destructive interference. In situations of demanding and degraded links (PRR ~ 10% with static PHY configurations), the algorithm sustained a PRR always higher than 98% without destructive interference and of at least 90% in the presence of destructive interference.

3. DETECTION OF SIDE-CHANNEL LEAKAGE

Security is a key aspect of dependability when facing not only environmental perturbation but an explicitly malicious environment. Security usually breaks down to (i) keeping sensitive information secret, and (ii) maintaining its integrity. For example, every Car-2-X communication link demands strong integrity guarantees in order to put trust into the exchanged data (e.g., vehicle positions, velocities). Likewise, coordinated platooning using global planning might expose vehicle positions and identities to network operators or even cloud providers, if resource intensive planning tasks are outsourced to the cloud. This demands not only confidentiality in the network, but also during computation. Integrity and confidentiality can be achieved by cryptographic algorithms with strong security guarantees. However, their concrete implementations often suffer from indirect information leaks, called side-channels, which can undermine all given guarantees.

3.1 Side-channel Attacks

Side-channel attacks range from mere time observation of network packets to sophisticated CPU attacks on the microarchitectural level. A whole class of these side-channel attacks exploits leakage within memory access patterns, that is, the address locations in memory a microprocessor or CPU accesses during computation. While such address-based leakage can be avoided by defensive coding paradigms, they are often not or improperly implemented in practice. Manually analyzing the side-channel resistance of software is both time consuming and error-prone.

In this work, we implement a fully-automated analysis framework for identifying arbitrary address-based leakage in cryptographic software. We

achieve this by (i) executing the program multiple times with differing input in a controlled environment and collecting its execution traces, (ii) analyzing the traces to find address differences, and (iii) correlating the found ones to the program input in case of non-deterministic program behaviour. Compared to existing work, we on the one hand precisely capture address leakage on a finest byte granularity for data leaks as well as for control-flow leaks. On the other hand, we explicitly address probabilistic programs, whose execution depends on some form of randomness. Using our tool, we analyze widely-used cryptographic libraries like OpenSSL and report known as well as previously unknown side-channel vulnerabilities.

In the following, we give a short background in Sect. 3.2, explain our methodology in Sect. 3.3, comment on non-determinstic program behaviour in Sect. 3.4 and conclude with results in Sect. 3.5.

3.2 Background

Address-based leakage occurs whenever a program's secret state is leaked through the memory addresses it accesses. This affects data memory as well as code memory. Consider the code snippet in Fig. 4, where different values of a `secret` input (line 2) access different table entries (line 1). If `secret` is `0`, entry `A` is accessed; for a `secret` value of `1`, entry `B` is accessed. Thus, the input leaks via the data addresses during table lookup, constituting a data leak.

```
1: int table[4] = { A, B, C, D };
2: int result = table[secret];
```

Fig. 4 – Data leakage.

Similar, the control-flow of a program can leak information about the secret. In Fig. 5, the function `process()` (line 2) is only executed if `secret` is `1` (line 1). This control-flow leak reveals one bit of `secret`.

```
1: if (secret == 1)
2:   process();
```

Fig. 5 – Control-flow leakage.

Whether a leak is exploitable in practice depends on the computing architecture and the precise attack methodology, which defines the granularity of observable leakage. While in the past strong assumptions were made about the possible granularity of observations, new attacks completely abrogated these assumptions [4]. Thus, in order to cover all address-based leaks, it is essential to operate on a finest byte granularity.

3.3 Methodology

Our approach to detect address leakage works in two phases: In the first phase, we execute the program, which is to be analyzed, multiple times with different secret input values, chosen at random. Execution is done in a controlled environment. We used a binary instrumentation framework to record all addresses (code and data) being accessed during program execution on a byte granularity. The addresses are stored in so-called address traces for later analysis.

Varying the secret input aims at triggering the vulnerable location in the program in order to show up in the address traces. Although this approach cannot guarantee exploring all possible control-flow paths of the program (i.e., no false negatives), we found that it is sufficient to detect a large number of actual leaks. Specifically, cryptographic algorithms tend to show strong diffusion of their input, which amplifies the explorative power of a small input set.

Other approaches in literature rely on symbolic execution with abstract interpretation of the processed data [5]. That is, the data plane is approximated to reduce otherwise exploding analysis times. While symbolic execution is good for maximizing coverage (i.e., minimize false negatives), approximation can cause analysis to wrongly report leaks where there are none (false positives). In contrast, our approach only detects actual address differences, thus avoiding false positives by design.

The second phase analyses the collected traces with respect to address differences. Since we operate in a controlled environment, we suppress any noise the operating system might induce on the memory addresses (e.g., address randomization techniques like ASLR). Hence, in order to find information leaks, it suffices to sequentially compare address traces. Any found difference constitutes a true information leak, assuming the analyzed program is purely deterministic. However, in practice, certain programs show non-deterministic behavior, which demands special treatment.

3.4 Detecting non-determinism

Many modern cryptographic algorithms rely on probabilistic methods to achieve certain security goals such as indistinguishability of cryptograms. When analyzing such implementations, their usage of randomness during the computation might spill over to the address traces as non-deterministic observation. That is, executing the program multiple times with one and the same input value can yield

differences in the address traces. To distill true leaks with a dependency on the secret input from non-determinism we first collect several execution traces under the same secret input. We then use the sequential trace comparison mentioned before to uncover non-deterministic leaks. Finally, we repeat the whole procedure for varying input values, as mentioned in Sect. 3.3 and subtract non-deterministic leaks in order to uncover true leaks.

3.5 Results & Discussion

We evaluated our analysis method on the widely-used OpenSSL library. We observed that most symmetric algorithms can be analyzed with already two executions. Only in one case were three traces necessary. Asymmetric algorithms of OpenSSL, on the other hand, require approximately 10 traces in order to achieve good coverage. We observed that a higher number of executions (we tested up to 30 traces) only uncovered very few additional leaks, which correspond to the same vulnerabilities we already found with fewer traces.

With our analysis, we re-confirmed known leakage in almost all symmetric ciphers of OpenSSL, which is due to a vulnerable lookup-table implementation. Moreover, we found previously unknown leakage in the initialization phase of RSA and DSA, which could allow an attacker to recover full cryptographic keys. We responsibly reported the vulnerabilities and provided appropriate patches to fix them in the source code.

4. PROTOCOL TESTING

In platooning, we see subsystems communicating at various levels to achieve some common goal. At a low level, networks of components inside vehicles communicate via bus systems. These components may, for example, control airbags. At a higher level, vehicles communicate among each other and with their environment via Car-2-X technology. All these forms of communication follow communication protocols, on which the involved parties agree. Since vehicles and their constituent components serve safety-critical functions, it is of utmost importance to ensure that they implement protocols correctly.

Therefore, we will discuss testing for functional correctness of communication protocols in the following. Protocol testing often uses as a basis finite state machines (FSMs) [6], which model expected responses to message transactions. In practice, this requires skilled engineers formalizing standards documents given in natural language. These documents often contain complex or even conflicting requirements, but may also leave scenarios unspecified. Protocol extensions and versions further add to the inherent complexity of such documents. This complexity makes complete formalizations difficult, labor-intensive, and error-prone.

Model-learning offers a solution to this problem by automatically learning formal models. In our work, we apply active automata learning [7,12], which learns FSM models via testing. In a case study, we applied this kind of learning for protocol testing in the context of the IoT [8]. Using the approach detailed in Sect. 4.2, we analyzed five implementations of MQTT [11], a publish-subscribe protocol for resource-constrained devices like IoT nodes.

4.1 Background

We will now briefly discuss the type of FSMs we use, and automata learning. Mealy machines serve as models in our work. These models have a finite number of states, inputs, and outputs. One of the states is designated as the initial state. The execution of a Mealy machine starts in the initial state. Upon the execution of an input, a Mealy machine changes its state and produces exactly one output. Fig. 6 shows an example of such a Mealy machine. It models parts of the behavior of the Mosquitto [13] MQTT broker. Its initial state is *s0* and transitions between states are labeled by inputs and corresponding outputs separated by slashes. For example, if we perform a connection input (*Con*) in s0, an acknowledgement output (*C_Ack*) will be produced and we move to state s1.

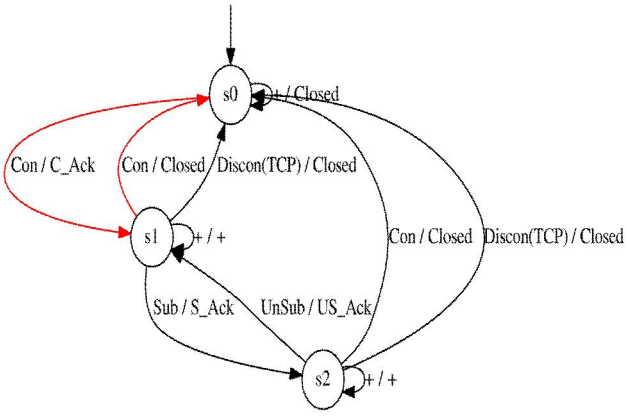


Fig. 6 – FSM model of the Mosquitto MQTT broker.

There are several approaches to learn such models from black-box systems. We applied active automata learning [7,12], which interacts with systems via testing. This style of learning usually iterates two phases. In an exploration phase, information is gathered via *membership queries*. Such a query executes a single input sequence and records the corresponding output sequence. Once we

have enough information to form a hypothesis model, we issue an equivalence query. In practice, this query tests for equivalence between the hypothesis and the black-box system. If we find an input sequence *s* demonstrating non-equivalence, we start with a new round of learning, exploring behavior related to *s*. Otherwise, we stop learning and output the last hypothesis as the learned model.

4.2 Learning-based Protocol Testing

The learning-based testing approach [8] that we discuss in the following, enables protocol testing without prior modeling of the desired behavior. Fig. 7 shows an overview of the approach, in which we examine pairs of protocol implementations and compare their observable behavior. The rationale behind this approach is that observable differences are likely to expose implementation errors.

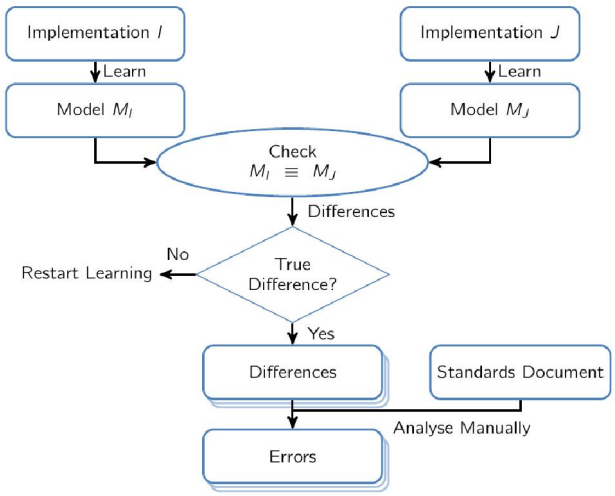


Fig. 7 – Overview of learning-based protocol testing.

First, we learn FSM models representing the implementations. Then, we check for observable equivalence between those models, i.e., we check whether they produce the same output sequences for all possible input sequences. During this procedure, we record all differences.

Each detected difference is further examined. First, we test the protocol implementations with the input sequence exposing the difference in model behavior. By that, we test whether this sequence leads to observably different behavior in the actual implementations. If such a test shows no observable difference, we know that at least one learned model is incomplete and we consequently extend it via more thorough learning. If a difference in model behavior corresponds to an actual difference, we check the standards document to determine whether the difference points to an implementation error. Basically, we perform *differential testing* on model level via equivalence checking and test systems only during learning and for analyzing differences.

The main benefit with respect to manual effort is that we do not need to formalize the complete standards document. It allows us to concentrate only on those parts that are implemented differently by various vendors. By following a bottom-up approach, from systems to models, we slightly change the objective of testing. We do not test against a standard, but we rather test whether two systems implement the standard in the same way. For this reason, the approach does not detect specification violations present in both examined systems. To detect a larger range of errors, we tested pairs of five systems in our case study.

4.3 Case Study: Testing MQTT Brokers

We investigated the behavior of five open-source implementations of MQTT brokers. These brokers are responsible for managing client connections, processing subscriptions, and for relaying published messages. Therefore, testing focused on various modes of connections and subscriptions, and types of messages. In the case study, we performed the following sequential steps:

1. We identified types of messages transmitted from clients to brokers.
2. For each of the types, we defined a corresponding abstract FSM input.
3. We grouped inputs with interdependencies among each other, forming seven groups of inputs.
4. For each of the groups and for each pair of brokers, we performed the learning-based testing approach discussed above.

Each group covers a certain aspect of MQTT, like processing invalid messages. This grouping helped to reduce model sizes, thereby reduce learning time while still covering a large portion of the MQTT functionality. Altogether, our experiments revealed 18 errors in four of the examined implementations.

One of the errors is highlighted in red in Fig. 6 and Fig. 8, which show models of the Mosquitto [13] and the HBMQTT [14] broker, respectively. They react differently if we try to connect again while being already connected. Mosquitto closes the network connection upon the second connection attempt. HBMQTT ignores this second connection attempt. According to the MQTT specification [11], HBMQTT behaves incorrectly – the network connection must be closed.

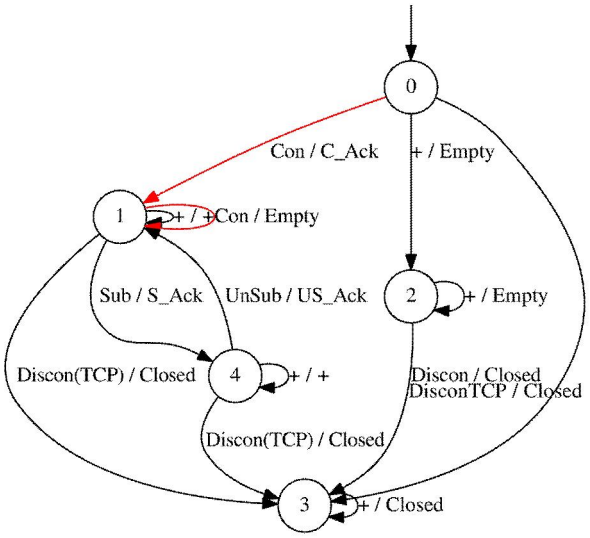


Fig. 8 – FSM model of the HBMQTT broker.

4.4 Discussion and Outlook

We were able to semi-automatically find errors in protocol implementations that are actively used. In addition to setting up the learning environment, we only needed to manually analyze the detected differences. While this shows that learning-based testing can effectively be applied, we also identified short-comings: (1) learning requires the execution of a large amount tests resulting in long execution times, and (2) FSMs cannot capture all relevant aspects, like stochastic and time-dependent behavior.

In subsequent work, we tackled the first issue by developing an equivalence test generation technique tailored toward learning [9]. This technique reduces the required number of test executions while still learning reliably. As a first step to mitigate the second issue, we developed a learning-based testing technique for stochastic systems [10]. Our current work focuses on learning timing behavior from tests. Ultimately, we plan to bring these techniques together to automatically test systems involving both stochasticity and time-dependent behavior.

5. NETWORKED CONTROL

5.1 Introduction

In the last decades conventional control systems are more and more replaced by networked feedback loops. The advantages of closing the control loop via (wireless) communication networks are increased flexibility and reduced wiring costs. Applications like automated driving or advanced production systems and robotics foster the development of methods for Networked Control Systems (NCS). The typical structure of an NCS is shown in Fig. 9. It consists of plant, sensors, actuators, the communication network, controllers and, if required, observers.

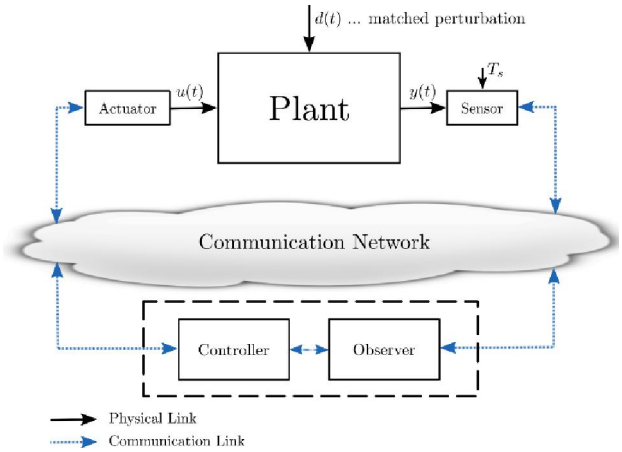


Fig. 9 – Structure of a Networked Control System.

Due to the heterogeneity of such systems, several disciplines need to cooperate in their design, see Fig. 10.

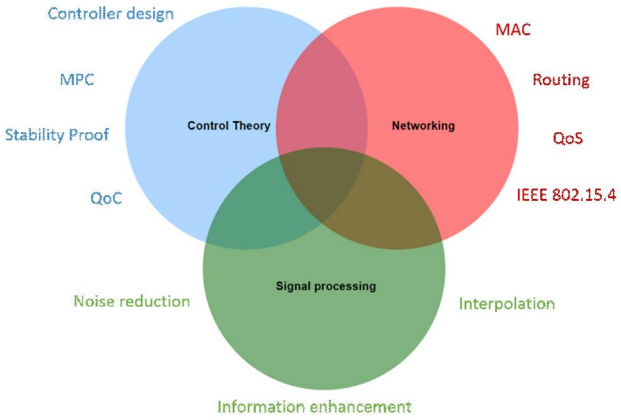


Fig. 10 – This figure shows how several aspects of control theory, networking and signal processing interact in the design of dependable networked control systems.

5.2 Challenges in Networked Control

The communication system renders analysis and design of NCS complex and makes it necessary to re-evaluate conventional control theories. Major challenges of NCS are the network impairments that can jeopardize the performance and stability of feedback loops. Among them are time varying delays and transmission intervals, packet dropouts, communication constraints and cyber attacks. The control algorithms should be able to cope with these uncertainties to allow dependable control of e.g. safety-critical systems. This is the motivation for the platooning scenario as described in Sect. 6, where the developed algorithms should be tested together

with the communication solutions provided in Sect. 2.

5.3 Sliding Mode Control for NCS

Sliding mode control (SMC) is a feedback control strategy, which allows to drive and then constrain the system state to a certain subspace of the state space (sliding manifold). Once the system state is on this surface, the closed loop response is insensitive to bounded perturbations occurring in the input channel of the plant (so called matched perturbations). However, SMC concepts are very sensitive to time delays leading to undesired dynamic phenomena and reduced control performance.

Based on the assumption that the sampling time is constant, a discrete time SMC algorithm is designed ensuring stability even in the presence of variable time delays. To overcome the fluctuations in the delay, we use buffering methods at the cost of an increased overall delay [15].

In a first step, a suitable discrete time model of the plant taking into account the constant buffer delay is derived. Then, a discrete time SMC algorithm based on the so-called reaching law approach is designed. It can be shown by simulation and experiment that the proposed control approach is superior to conventional SMC techniques. Experimental data can be found in [15]. There it is shown that the design of an SMC without considering the uncertainties of the communication system might even lead to instability of the closed loop system. With the approach proposed in [15] it is possible to guarantee a stable closed loop behaviour even if time varying delays are present.

From a practical point it could be regarded as a disadvantage that all state variables have to be measureable. This drawback can be overcome by using suitable state observers.

5.4 Robust Observers for NCS

So-called Unknown Input Observers (UIO) are capable of estimating the plant states correctly despite the presence of matched perturbations. Even more, these perturbations can be reconstructed. However, in the case of measurement dropouts, the functionality of UIOs cannot be guaranteed anymore. This motivates research in the field of UIO design for NCS.

We are working on novel concepts, which allow the use of UIOs in a networked environment. In a first step, only dropouts in the communication channel connecting sensor and observer are taken into account. A simple stochastic model for modeling the lossy transmission channel is the so-called Gilbert-Elliot Markov Chain model, see Fig. 11.

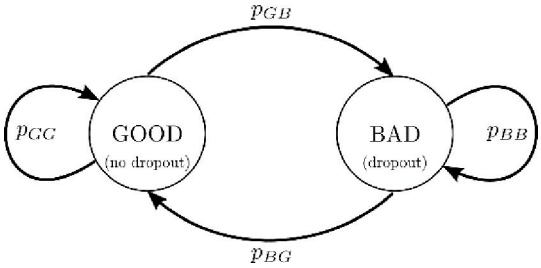


Fig. 11 – Gilbert-Elliot model.

Based on this dropout model, an UIO with guaranteed stability properties up to a certain dropout probability can be designed. The developed concept shows promising performance in simulation and experiment.

5.5 Outlook

Future research will focus on the combination of the two above presented NCS methods. The respective algorithms will be extensively evaluated in the Truck Testbed presented in Sect. 6.

6. TRUCK TESTBED

In order to investigate the concepts discussed in the previous sections, a testbed with small-scale vehicles (see Fig. 12) for automated driving is used. The following requirements on the testbed are fulfilled:

- scalability: the testbed must be scalable so that multiple vehicles can be used
- distribution: the software is implemented on embedded systems mounted on trucks
- real-time capability: the execution times of different tasks must be ensured
- flexibility: easy and affordable changes of hardware and software are possible.

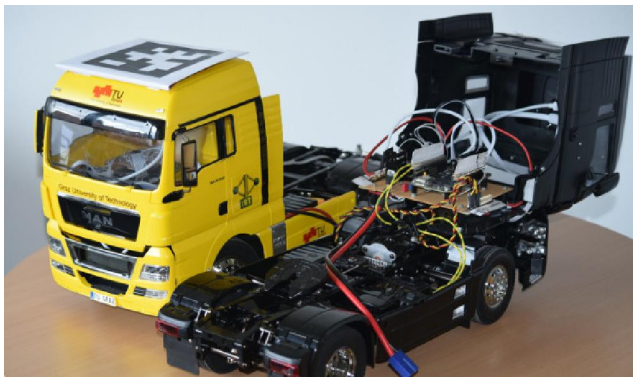


Fig. 12 – The model trucks are equipped with a BeagleBone Black board to test different scenarios on the testbed.

For platooning applications, small-scale trucks in scale 1:14 [16] have been built up. The setup of the testbed is shown in Fig. 13, where the relation between hardware and software is depicted. These components are described subsequently.

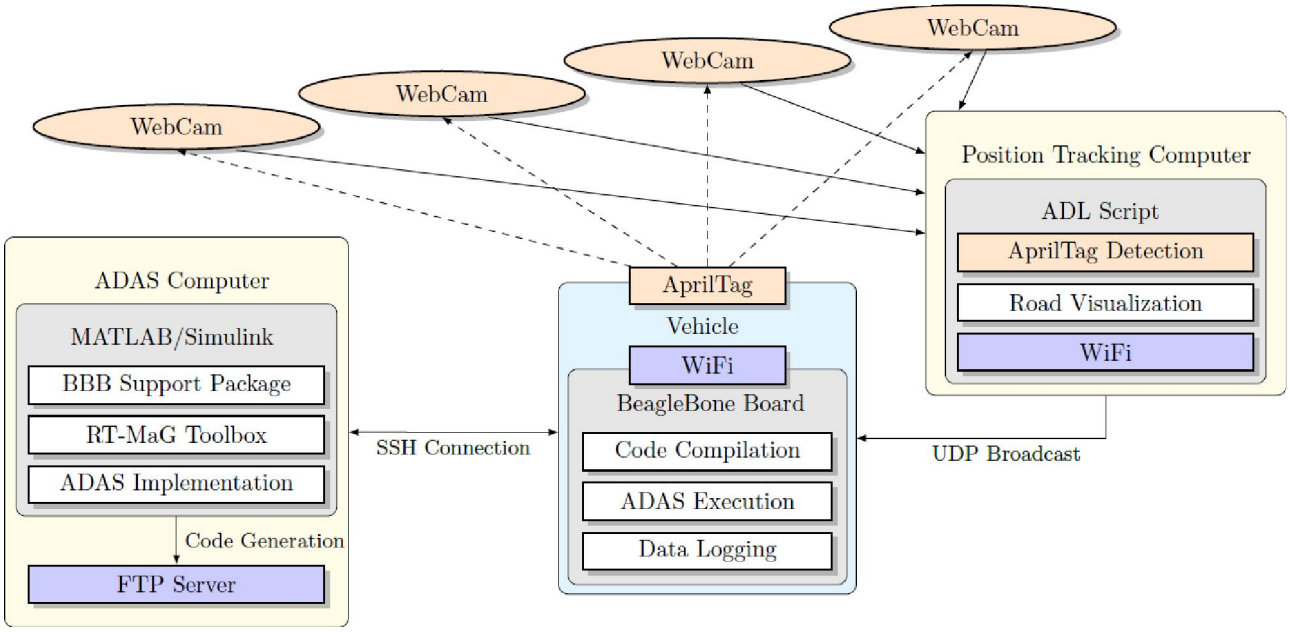


Fig. 13 – Setup of the testbed using small-scale vehicles and communication.

6.1 Small-scale Vehicles

Each vehicle is equipped with a BeagleBone Black board [17], which executes different advanced driver assistance systems (ADAS). These ADAS functionalities can be modeled in MATLAB/Simulink and code is generated automatically on an ADAS Computer using appropriate MATLAB packages [18, 19]. The code is then downloaded and compiled on the BeagleBone Black (using a real-time capable operating system [20]), and the execution of the ADAS on this on-board computer is triggered via SSH connection.

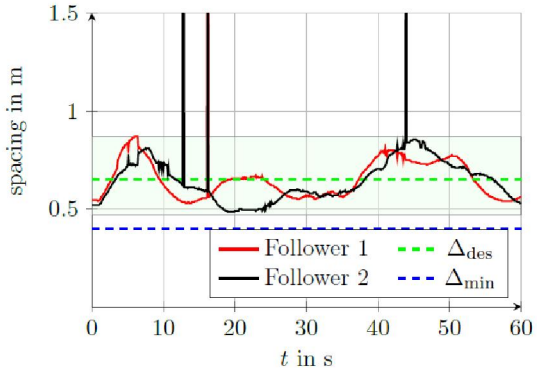
6.2 Position Tracking and Communication

For motion planning in automated driving, the positions of the vehicles need to be measured. For this purpose, an inexpensive and easy to use indoor localization was built up using off-the-shelf webcams [21]. The positions and orientations of the trucks are detected via AprilTags [22] that are mounted on top of the vehicles. The position tracking code is based on the open source AprilTags C++ Library [23]. The tag detection algorithm is executed on the Position Tracking Computer as indicated in Fig. 13. The road coordinates are stored on the trucks and can be displayed additionally on the images of the webcams on this computer. The detection algorithm calculates the vehicles' positions with a rate of 10Hz and broadcasts the information via WiFi using UDP. An accuracy of approximately 0.03m can be achieved, which is sufficient for many tests.

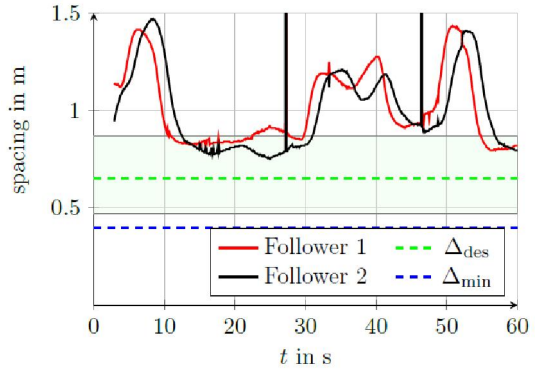
In addition to the position data, environmental or tactical information can be sent via UDP broadcast to the vehicles, emulating Car-to-Infrastructure (C2X) communication. Moreover, the trucks can receive UDP packets from other vehicles, i.e., Car-to-Car (C2C) communication is possible. Note that automated driving functionalities such as Platooning rely on communication between the trucks.

6.3 Platooning

Platooning is currently the focus of cooperative automated driving, see, e.g., [24], [25]. In platooning scenarios, the aim is to decrease the air drag of several trucks in order to reduce fuel consumption. For this purpose, the trucks maintain very small distances with respect to the preceding vehicle, which also increases the capacity of the roads. In order to guarantee safety for small intervehicle distances, communication between the vehicles is necessary. The results of the platooning experiment on the testbeds are shown in Fig. 14. If the Car-2-car communication can be ensured as in a), maintaining very small inter-vehicle distances is possible. However, if no communication is available, the distances must be much larger as in b) due to the delayed reaction that can be observed in the velocities.



a) constant distance spacing with C2C communication



b) constant time-headway spacing without C2C communication

Fig. 14 – Inter-vehicle spacings of the two following trucks. C2C communication allows for smaller distances while still guaranteeing safety. Note that the peaks in the distances arise due to sensor faults.

7. CONCLUSION

In order to enable safety-critical applications of the IoT, substantial research is needed to bring dependability to the IoT. In this paper we have presented recent research towards dependable communication, security, protocol verification, and networked control. Ultimately, these contributions will be integrated into a common framework to facilitate giving performance guarantees also in harsh environments, thus also enabling an informed decision if a given IoT application matches the safety requirements of the application.

8. ACKNOWLEDGEMENTS

This work has been performed within the LEAD project "Dependable Internet of Things in Adverse Environments" funded by Graz University of Technology. The authors want to thank M. Rotulo and A. Luppi for their contributions to the section on networked control.

REFERENCES

[1] B. Kempke, et al., "SurePoint: Exploiting ultra wideband flooding and diversity to provide robust, scalable, high-fidelity indoor localization," *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*, 2016.

[2] S. Marano, W. M. Gifford, H. Wymeersch and M. Z. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE Journal on Selected Areas in Communications*, Vol. 28, No. 7, pp. 1026-1035, 2010.

[3] Decawave, [Online]. Available: http://www.decawave.com.

[4] Y. Yarom, D. Genkin and N. Heninger, "CacheBleed: a timing attack on OpenSSL constant-time RSA," *Journal of Cryptographic Engineering*, Vol. 7, No. 2, pp. 99-112, 2017.

[5] G. Doychev, B. Köpf, L. Mauborgne and J. Reineke, "CacheAudit: A tool for the static analysis of cache side channels," *ACM Transactions on Information and System Security*, Vol. 18, No. 1, pp. 4:1-4:32, 2015.

[6] G. V. Bochmann and A. Petrenko, "Protocol testing: review of methods and relevance for software testing," *Proceedings of the 1994 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 109-124, 1994.

[7] B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," *Lecture Notes in Computer Science*, Vol. 6659, pp. 256–296, 2011.

[8] M. Tappler, B. K. Aichernig, and R. Bloem, "Model-based testing IoT communication via active automata learning," *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation, ICST 2017*, pp. 276–287, 2017.

[9] B. K. Aichernig, M. Tappler, "Learning from faults: Mutation testing in active automata learning," *Lecture Notes in Computer Science,* Vol. 10227, pp. 19–34, 2017

[10] B. K. Aichernig, M. Tappler, "Probabilistic black-box reachability checking," *Lecture Notes in Computer Science,* Vol. 10548, 2017.

[11] Edited by Andrew Banks and Rahul Gupta, "MQTT Version 3.1.1. OASIS Standard", October 2014.

[12] D. Angluin, "Learning regular sets from queries and counterexamples," *Inf. Comput.*, Vol. 75, No. 2, pp. 87–106, 1987.

[13] Mosquitto, [Online]. Available: https://mosquitto.org/.

[14] HBMQTT, [Online]. Available: https://github.com/beerfactory/hbmqtt.

[15] J. Ludwiger, et. al., "Towards networked sliding mode control," *Proceedings of the 56th*

IEEE Conference on Decision and Control, 2017.
[16] Tamiya Modelltrucks, [Online]. Available: http://www.tamiya.de/de/produkte/rcmodelltrucks.htm.
[17] BeagleBone Black Board, [Online]. Available: https://beagleboard.org/black.
[18] MATLAB Support Package for BeagleBone Black Hardware, [Online]. Available: https://de.mathworks.com/help/supportpkg/beagleboneio/.
[19] A. Manecy, N. Marchand, and S. Viollet. "RT-MaG: An open-source SIMULINK toolbox for linux-based real-time robotic applications," *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014.
[20] RT PREEMPT patch for BeagleBone, [Online]. Available: http://elinux.org/BeagleBoardDebian#Mainline_.284.4.x_lts.29.
[21] Logitech WebCam C930e, [Online]. Available: http://www.logitech.com/de-at/product/c930e-webcam
[22] E. Olson. "AprilTag: A robust and flexible visual fiducial system," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.
[23] AprilTags C++ Library, [Online]. Available: http://people.csail.mit.edu/kaess/apriltags/
[24] European truck platooning challenge, [Online]. Available: https://www.eutruckplatooning.com.
[25] A. Alam, J. Mårtensson, and K.H. Johansson. "Experimental evaluation of decentralized cooperative cruise control for heavy-duty vehicle platooning," *Control Engineering Practice*, Vol. 38, pp. 11–25, 2015.

Bernhard Großwindhager *received the B.Sc. and Dipl.-Ing. degrees in Electrical Engineering from Graz University of Technology, Austria, in 2012 and 2014, respectively. Since 2016 he is a PhD student at the Networked Embedded Systems group of the Institute for Technical Informatics at TU Graz. His research interests include reliable and efficient ultra-wideband wireless communication and localization.*

Astrid Rupp *is a PhD student and project assistant at the Institute of Automation and Control at Graz University of Technology, Austria. She received the master's degree in Information and Computer Engineering at Graz University of Technology in 2013. Her research focuses on automated driving, multi-agent systems, formation control, and sliding mode control.*

Martin Tappler *is a PhD student and project assistant at the Institute of Software Technology at Graz University of Technology, Austria.*

He received a bachelor's and a master's degree in computer science from Graz University of Technology. His research focuses on model-based testing, test-based model learning, and combinations thereof.

Markus Tranninger *is a research assistant at the Institute of Automation and Control at Graz University of Technology, Austria.*

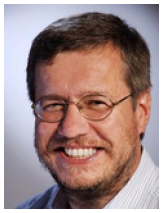
His research interests include networked control, robust control and estimation, and real-time co-simulation.

Samuel Weiser *is a PhD student at the Secure Systems group at Graz University of Technology, Austria. He received his master's degree in Information and Computer Engineering at TU Graz in 2016. His research focuses on software side-channels and secure computing architectures.*

Bernhard K. Aichernig *is a tenured associate professor at Graz University of Technology, Austria. He investigates the foundations of software engineering for realising dependable computer-based systems. Bernhard is an expert in formal methods and testing. His research covers a variety of areas combining falsification, verification and abstraction techniques. Current topics include the Internet of Things, model learning, and statistical model checking. Since 2006, he participated in four European projects. From 2004-2016 Bernhard served as a board member of Formal Methods Europe, the association that organises the Formal Methods symposia. From 2002 to 2006 he had a faculty position at the United Nations University in Macao S.A.R., China. Bernhard holds a habilitation in Practical Computer Science and Formal Methods, a doctorate, and a diploma engineer degree from Graz University of Technology.*

Carlo Alberto Boano *is an assistant professor at the Institute for Technical Informatics of Graz University of Technology, Austria. He received a doctoral degree sub-auspiciis praesidentis from TU Graz in 2016 with a thesis on dependable wireless sensor networks. Carlo Alberto's research interests encompass the design of dependable networked embedded systems, with emphasis on the energy-efficiency and reliability of low-power wireless communications, as well as on the robustness of networking protocols against environmental influences.*

Martin Horn *is professor at and director of the Institute for Automation and Control at Graz University of Technology, Austria. His research interests encompass networked control, variable structure control, modeling of mechatronic systems and automotive applications.*



Gernot Kubin *was born in Vienna, Austria. He received the Dipl.-Ing. degree in 1982 and the Dr.techn. degree (sub auspiciis praesidentis) in 1990 in Electrical Engineering from Vienna University of Technology, Austria. He is a Professor of Nonlinear Signal Processing and has been Head of the Signal Processing and Speech Communication Laboratory at Graz University of Technology, Austria, since 2000. At TU Graz, he has been Dean of Studies in Electrical and Audio Engineering 2004-2007, Coordinator of the Doctoral School in Information and Communications Engineering since 2007, and Chair of the Senate 2007-2010 and again since 2013. His research interests are in nonlinear signals and systems, computational intelligence, as well as speech and audio communication. Dr. Kubin has been a Member of the Board Austrian Acoustics Association (since 2000), an elected member of the Speech and Language Processing Technical Committee of the IEEE (since 2011), an elected member of the Speech Acoustics and Speech Processing committees of the German Information Technology Society ITG (since 2015).*

Stefan Mangard *is professor at Graz University of Technology, where he heads the Secure Systems group. His research interests include security architectures, software side-channels, hardware attacks and countermeasures, cryptography as well as secure and efficient hardware and software implementations of cryptography. He obtained an ERC consolidator grant in 2015 to research countermeasures in hardware and software to protect software execution against all types of side-channel and fault attacks.*

Martin Steinberger *was born in Leoben, Austria. He received the master in electrical engineering and the PhD in technical sciences from Graz University of Technology, Graz, Austria, in 2005 and 2011, respectively. He has been assistant professor at the Institute of Automation and Control since 2016, Graz University of Technology. His research is mainly focused on sliding mode control and networked control.*

Kay Römer *is professor at and director of the Institute for Technical Informatics at Graz University of Technology, Austria. He received a PhD in computer science from ETH Zurich, Switzerland in 2005. His research interests encompass wireless networking, fundamental services, operating systems, programming models, dependability, and deployment methodology of networked embedded systems.*