Abstract

In this deliverable, we report on the results of Work Package 3 (Dependable Networking) obtained in the first year of the WSAN4CIP Project. These results are related to the identification of the design principles of dependable networking mechanisms for WSANs. In our work, and hence, in this deliverable, we follow the layered model of networking protocol stacks: We identify the most important dependability concepts and models at the physical, MAC (Medium Access Control), routing, and transport layers, and we analyze existing networking protocols from the different layers proposed in the literature with respect to the identified dependability properties.

[End of abstract]
Disclaimer

This document contains material, which is the copyright of certain WSAN4CIP consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):
All WSAN4CIP consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the WSAN4CIP consortium as a whole, nor a certain party of the WSAN4CIP consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Short project title] WSAN4CIP
[Number and title of work-package] WP3 Dependable Networking
[Document title] Dependability concepts, models, and analysis of networking mechanisms for WSANs
[Editor: Name, company] Levente Buttyan, Budapest University of Technology and Economics
[Work-package leader: Name, company] Levente Buttyan, Budapest University of Technology and Economics
[Estimation of PM spent on the Deliverable] 14 PM

Copyright notice
© 2009 Participants in project WSAN4CIP
## List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>BME</td>
<td>Levente Buttyan</td>
</tr>
<tr>
<td>BME</td>
<td>Gergely Acs</td>
</tr>
<tr>
<td>BME</td>
<td>Peter Schaffer</td>
</tr>
<tr>
<td>BME</td>
<td>Karoly Farkas</td>
</tr>
<tr>
<td>BME</td>
<td>Boldizsar Bencsath</td>
</tr>
<tr>
<td>BME</td>
<td>Ta Vinh Thong</td>
</tr>
<tr>
<td>BME</td>
<td>Aron Laszka</td>
</tr>
<tr>
<td>INOV</td>
<td>Antonio Grilo</td>
</tr>
<tr>
<td>NEC</td>
<td>Alban Hessler</td>
</tr>
<tr>
<td>LTU</td>
<td>Laurynas Riliskis</td>
</tr>
<tr>
<td>LTU</td>
<td>Evgeny Osipov</td>
</tr>
<tr>
<td>INRIA</td>
<td>Daniele Perito</td>
</tr>
<tr>
<td>INRIA</td>
<td>Claude Castelluccia</td>
</tr>
</tbody>
</table>

© WSAN4CIP consortium 2009
5 Dependability of MAC protocols

5.1 Notions of reliability

5.2 Notions of security

5.3 Analysis of selected MAC protocols

5.3.1 S-MAC
5.3.2 T-MAC
5.3.3 DS-MAC
5.3.4 PMAC
5.3.5 TRAMA
5.3.6 FLAMA
5.3.7 WiseMAC
5.3.8 B-MAC
5.3.9 X-MAC
5.3.10 MFP-MAC
5.3.11 DEEJAM
5.3.12 Dragon-MAC

5.4 Summary and outlook

5.5 Future Work

6 Dependability at the physical layer

6.1 Jamming attack model and metrics

6.1.1 Adversarial model
6.1.2 Success metrics
6.1.3 Jamming attacks on sensor networks

6.2 Anti-jamming techniques

6.2.1 Spread spectrum techniques
6.2.2 Other anti-jamming techniques for WSNs

6.3 Summary and outlook

7 On the robustness of network topologies

7.1 Existing robustness metrics

7.2 Comparison of existing metrics

7.3 Future work
Executive summary

The usefulness of Wireless Sensor and Actuator Networks (WSANs) in Critical Infrastructure Protection (CIP) applications is primarily determined by the dependability of the WSAN itself. In this context, dependability means that the WSAN provides its services with reasonable quality in all reasonably conceivable circumstances such that CIP applications can really depend on those services. These conceivable circumstances usually include both accidental failures and intentional attacks, which are usually addressed by reliability and security mechanisms, respectively, integrated into the WSAN architecture and protocols. Hence, dependability for us is a term that covers both reliability and security aspects together.

In this deliverable, we report on the results of Work Package 3 (Dependable Networking) obtained in the first year of the WSAN4CIP Project. One of the main objectives of this work package is to identify and study the design principles of dependable networking mechanisms for WSANs. For this, it is indispensable to understand dependability in a finer granularity level and in the context of particular networking functions, such as routing, transport, etc. The structure of our work, and accordingly, this deliverable, follows the layered model of networking protocol stacks: we identify the most important dependability concepts and models at the physical, MAC (Medium Access Control), routing, and transport layers, and we analyze existing networking protocols from different layers proposed in the literature with respect to the identified dependability properties. More specifically, the organization of the material in this deliverable is the following:

• In Section 1, we study WSN transport protocols. The main objective of these protocols is to ensure reliable data delivery on an end-to-end level; however, our analysis shows that most existing WSN transport protocols achieve reliability only in a benign environment, but they fail in face of an attacker that can forge and inject control packets, such as acknowledgements or negative acknowledgements. We argue that cryptographic protection of control packets is needed in order to prevent their forgery. While such cryptographic protection can be provided at layers below the transport layer, we believe that that approach would not be efficient, hence we identify the need for a cryptographically secured WSN transport protocol.

• In Section 2, we study a particular transport problem: the reliable dissemination of large size data (e.g., a program image) to a set of sensor nodes in the WSAN. We identify the design requirements for a multicast data dissemination protocol, including reliability and security, and we review the state-of-the-art of such protocols. We identify fountain codes and network coding as a promising approach to ensure the reliability of data dissemination in WSNs. We argue, however, that the problem of pollution attacks against network coding schemes must be properly addressed in order to make distributed coding based data dissemination not only reliable, but also dependable. We sketch our approach that we intend to follow in the project to solve this problem.

• In Section 3, we study various aspects of WSN routing protocols with an emphasis on dependability issues. Routing in WSANs is an extensively studied problem in the literature, and there are a huge number of routing protocols proposed. Instead of creating yet another survey of those protocols, we adopted here a different approach: we attempt to factor out the main design principles for WSN routing, as well as to identify the most important dependability concepts in this context. For this, we define a modular approach where various components responsible for different aspects of routing can be combined together to obtain a specific routing protocol with desirable properties. This also requires us to define network and operational models, as well as objectives for routing protocols. We give detailed description of a handful of routing protocol components, including those responsible for reliability and security properties, and we refer to state-of-the-art routing protocols where those components are implemented. We advocate the modular approach of combining existing components in order to create new routing protocols for WSANs that have the desired dependability properties; this is the approach that we intend to follow in the project.
In Section 4, we identify dependability properties of WSN clustering and cluster head election protocols, and we analyze state-of-the-art cluster head election protocols with respect to those properties. Our analysis shows that, similar to transport protocols, cluster head election protocols proposed in the literature do not satisfy even the basic properties required for dependability if they are faced with an attacker that can actively interfere with the execution of the protocol.

Section 5 deals with dependability at the MAC (Medium Access Control) layer. After identifying the most important properties of reliability and security in this context, we briefly analyze some state-of-the-art MAC protocols proposed for WSANs. We conclude that while reliability aspects are well-covered (which is not surprising considering that one of the main objectives of MAC protocols is to ensure reliable data transfer through a wireless link), security aspects are usually not considered at this layer.

Section 6 is about the problem of jamming attacks. We define attacker models for and metrics to measure the success of jamming attacks, and we give an overview of state-of-the-art anti-jamming approaches and techniques for WSANs.

Finally, Section 7 is concerned with the problem of measuring the robustness of network topologies. We argue here that the most well-known metric, based on the notion of graph connectivity, has some limitations, and we give a brief overview of some alternatives, such as graph toughness and graph strength. This topic requires more theoretical work aiming at defining a robustness metric that fits better the WSAN network model, and that can serve as the basis of node deployment strategies.
1 Dependability of transport protocols

There are applications of WSNs where the sensors capture and transmit high-rate data (e.g., multimedia sensor networks [4]). In those applications, special mechanisms are needed to ensure end-to-end reliability and congestion control. Such mechanisms are usually implemented in the transport layer of the communication stack in form of a transport protocol. It is widely accepted that transport protocols used in wired networks (e.g., the well-known TCP) are not applicable in wireless sensor networks, because they perform poorly in a wireless environment and they are not optimized for energy consumption. For this reason, a number of transport protocols specifically designed for WSNs have been proposed in the literature (see [157] for a survey). The main design criteria that these protocols try to meet are the following:

- **Reliability**: WSNs often suffer from link quality problems, which, along with congestions and collisions, can lead to high packet loss ratios. Therefore, in order to ensure reliable communication, WSN transport protocols must implement some packet loss detection and retransmission scheme.

- **Congestion control**: Due to the large number of nodes and the low bandwidth, congestions can occur, especially at nodes close to the sink. Also, in case of event driven applications when an event occurs, a large burst of packets is generated that can lead to high collision rate. As congestion and packet collision can greatly degrade the overall performance of the system, these questions must be taken into account in WSN transport layer protocols.

- **Energy efficiency**: Sensor nodes have small memory and computational capability, along with low communication bandwidth and limited battery lifetime. Therefore, in order to maintain long network lifetime, WSN transport protocols must be lightweight and must keep communication overhead as low as possible.

Interestingly, despite the fact that WSNs are often envisioned to operate in hostile environments, none of the proposed WSN transport protocols address security issues. As a consequence, the proposed protocols meet the above requirements only in a benign environment, but they fail in a hostile environment. In particular, our analysis shows that most of the proposed WSN transport protocols fail to provide end-to-end reliability and are subject to increased energy consumption in the presence of an adversary that can perform active attacks on the communication channels.

We must also note that there are many papers on security issues in WSNs (see e.g., [101] and the references therein), but to the best of our knowledge, the security of the transport layer in WSNs has been neglected so far. Indeed, most of the literature on WSN security deals with MAC layer and network layer security issues, and key management problems. In contrast to those works, in this section of this report, we focus on the security issues at the transport layer.

The rest of the section is organized as follows: In Subsection 1.1, we give a high level summary of the various acknowledgement schemes used to provide end-to-end reliability. In Subsection 1.2, we specify our attacker model and define metrics required to measure the impact of an attack. In Subsection 1.3, we describe specific WSN transport protocols and analyze them with respect to dependability. Finally, in Subsection 1.4, we summarize the lessons that we learnt from this analysis.

1.1 Transport layer reliability mechanisms

Communications in WSNs usually take place between the sensor nodes and the base stations, and it is important to distinguish the direction of those communications. In case of upstream communication, the sender is a sensor node, and the receiver is a base station, while in case of downstream communication, these roles are reversed.

The goal of the sender is to reliably transmit to the receiver a full message that may consist of multiple fragments. If a fragment is lost, it must be retransmitted. This may be done in an end-to-end manner, where the source node itself repeats the lost fragment, or on a hop-by-hop basis, where intermediate nodes can cache and retransmit fragments if they are lost.
A reliable protocol can only detect fragment losses, if there is some kind of feedback in the system. Typically the following types of feedbacks are used:

- **Acknowledgement (ACK):** Acknowledgements can be:
  - **Explicit** – Upon receiving a fragment the node sends back a confirmation on it. An explicit ACK can confirm the reception of a single or multiple fragments.
  - **Implicit** – When a node overhears his neighbor forwarding a fragment sent by the node, it can assume that the delivery of the fragment to that neighbor was successful. This method can only confirm the delivery of a single fragment.

- **Negative Acknowledgement (NACK):** If a node somehow becomes aware of the fact that it did not receive a fragment, it can explicitly send a request for retransmission. A NACK can also refer to a single or multiple requested fragments. In multiple NACK schemes the notion of loss window refers to a range of lost fragments.

- **Selective Acknowledgement (SACK):** It is a combination of an explicit single or multiple ACK – used for the last fragments received in-order – and multiple ACKs for other fragments that were also received, but which are out of order.

Finally, we should mention the following two important theoretical problems related to NACK based schemes:

- **Lost last fragment problem:** Most NACK-based protocols use sequence numbers to detect fragment losses. If a node receives a fragment with a sequence number higher than expected, then it concludes that a fragment is lost. However, this method cannot detect if the last fragments of a stream are lost, since they will not going to be followed by a fragment with a higher sequence number. NACK based schemes must implement a specific solution for this problem.

- **Lost full message problem:** In wireless networks, it is possible, that an entire message is lost during transmission, as losses often occur in bursts, and messages in WSNs tend to consist of a few (often only one) fragments. Loss of an entire message cannot be directly detected by NACK based schemes, as the receiver never becomes aware of the existence of the message. This problem also requires special handling in NACK based schemes.

### 1.2 Attacker model

We assume that the attacker can eavesdrop the communications between any two nodes in the network, and she can forge and inject control packets anywhere in the network with a specified transmit power. However, we do not allow an attacker to delete (jam) packets, or to modify their content. Of course, we do understand that such attacks are possible in wireless networks, but if we allowed them in our attacker model, then no transport layer protocol would be able to ensure the end-to-end reliability of the communications [85]. In other words, we assume here that packet deletion and modification attacks are addressed at lower layers in the communication stack.

Our attacker model is not affected by security mechanisms applied at the application layer, because we are not interested in the content of the data packets, and the attacker in our model only injects transport layer control packets. In contrast to this, security mechanisms implemented below the transport layer (e.g., link layer packet authentication) would be useful to prevent some of the attacks, but in fact, none of the proposed transport protocols assume any security mechanisms at lower layers, and therefore, we will not assume their presence either in our analysis.

Attacks against WSN transport layer protocols come in two flavors:

- **Attacks against reliability:** Reliability in the context of transport protocols refers to reliable data transfer. In particular, a reliable transport protocol must be able to guarantee that every fragment loss is detected and that lost fragments can be retransmitted until they reach their destination. Thus, an attack against reliability is considered to be successful, if either a fragment loss remains undetected, or the attacker can permanently deny the delivery of a fragment.
• **Energy depleting attacks**: Energy depleting attacks are unique to sensor networks. In this case the goal of the attacker is to force the sensor nodes to perform energy intensive operations, in order to deplete their batteries, and thus, to decrease the lifetime of the network. In WSNs, the overall energy consumption of a sensor node is highly proportional to the number of the packets transmitted by the node. Therefore, an energy depleting attacker may try to coerce the sensor nodes to re-transmit fragments. In this case, we measure the successfulness of an attack by the ratio between the number of packets injected by the attacker in the network and the overall number of packets sent by the sensor nodes due to those injected packets.

1.3 Analysis of existing transport protocols

1.3.1 PSFQ

PSFQ (Pump Slowly, Fetch Quickly) [156] is an important general-purpose transport protocol, that provides downstream reliability with hop-by-hop recovery. The name of the protocol originates from the delivery method it uses. During the transmission, data fragments are transferred (pumped) with a relatively small speed, but if an error is detected, the protocol tries to quickly recover (fetch) the missing fragments from immediate neighbors.

**Protocol overview:** In PSFQ, every message is broadcasted, as the protocol assumes that each message must be transmitted to every node. If a specific node needs to be addressed, PSFQ can work on the top of an existing routing protocol. PSFQ uses a multiple NACK-based scheme to achieve reliability. It has three different working modes:

- **Pump operation:** This mode is responsible for the normal data transfer. Each data fragment has four fields: file ID, file length, sequence number and a TTL (Time To Live) value. In this mode, the source broadcasts a data fragment every $T_{\text{min}}$. If a node receives a fragment, it checks its local data cache and discards any duplicates. PSFQ buffers every new fragment, decreases their TTL value and schedules them to forward, if there are no gaps in the sequence numbers and TTL is not zero. Each scheduled fragment is delayed for a random period between $T_{\text{min}}$ and $T_{\text{max}}$ before it is forwarded. Within this random period, the node counts the number of times the same fragment is heard from neighboring nodes. If this counter reaches 4 before the scheduled rebroadcast, the transmission is canceled.

- **Fetch operation:** A node goes into fetch mode once a gap is detected in the sequence numbers. PSFQ uses NACKs with three header fields: file ID, file length and loss windows. Each node attempts to obtain all lost fragments in a single fetch operation. To reduce collisions, neighbor nodes wait a random time before transmitting missing fragments. Other nodes that have the same missing fragment will cancel their scheduled retransmission if they hear a repair for the same fragment. NACKs are aggressively repeated for non-received fragments. However, NACK packets are only propagated once, and only after the number of repetition for the same NACK exceeds a predefined threshold. To tackle the lost last fragment problem, each node can enter fetch mode pro-actively and send a NACK for the next missing fragment of the remaining fragments after a period of time.

- **Report operation:** This working mode was designed to feedback data delivery status information, however it has marginal influence on our the security analysis.

**Weaknesses:** One important problem with PSFQ is that it does not deal with the lost full message problem. Although the authors refer to this problem in the proactive fetch section, they do not provide a solution. This indicates that the protocol is not reliable, especially if the implementing WSN application uses relatively small messages consisting only of a few fragments.

Another general problem of the protocol is the inappropriate handling of TTL values. Due to the randomized transfer delays used by the forwarding method, it is possible that one fragment arrives to a node from the source earlier on a longer path than on the shortest one. This specific fragment will be
scheduled for forwarding, even if it has a smaller TTL value than other fragments for the same file ID. If fragments arriving later, but with higher TTL value are discarded due to collision, it is possible that the destination will not going to receive this fragment as it will be dropped earlier. Also, an attacker can inject a fragment with a given file ID, file size and sequence number but with a TTL value as low as 1. This fragment might prevent proper propagation of the valid fragment – as it will be discarded – which can lead to permanent fragment losses. If duplicated fragments with higher TTL values are also propagated and NACKs can force retransmission of fragments with zero TTL, this problem can be corrected, but it will not going to be efficient.

Reliability can be further compromised by an attacker using the communication model showed as an example in Figure 1: After node $B$ receives a fragment from $A$, it waits some time before forwards it to $C$. If the attacker $M$ can convince $B$ that four other nodes have already forwarded the same fragment before the timer expires, $B$ will drop the fragment. It can be easily done, as the attacker only needs to send the same fragment to $B$, each time spoofing a different source. If only $B$ receives these fraud fragments – which is ensured by the shortened radio range of node $M$ – the attack remains undetected.

![Figure 1: Packets injected by node $M$ are received only by node $B$ due to the shortened radio range of $M$](image)

Even if $C$ can later detect this fragment loss and can broadcast a NACK for the fragment, with the similar technique, it is easy to prevent $B$ to respond to the NACK. The attacker only needs to immediately inject a false response to the NACK that only $B$ receives. With these methods, it is possible to permanently delete arbitrary messages from the system, making it unreliable.

Energy depleting attacks are also possible against PSFQ. In general, an injected fake NACK forces a WSN to unnecessarily replay a fragment. For multiple NACK schemes one packet can provoke the retransmission of multiple fragments, which multiplies the impact of the attack. The problem with PSFQ, is that it does not limit the size of the loss window, so it can be as high as the size of the largest transmitted data. Similarly, an attacker can inject a fragment with a large sequence number, which generates large loss windows in multiple nodes. Even with non-propagating NACKs and hop-by-hop recovery, these attacks have the impact of $O(s)$, where $s$ is the number of packets required to transmit a data fragment. Note, that for small $s$ values, the reliability of the protocol is low, due to the unresolved lost full message problem.

We note, that another beneficial attack against the protocol would be to inject a false fragment with a new file ID, file length of 2, sequence number of 1 and a TTL value as high as possible. Since this is the first fragment of a file, it does not generate a gap in the sequence numbers, consequently every node will immediately propagate the fragment and due to the high TTL value, it will reach every node. As only the last fragment is missing from the entire message, every node, that receives it, will pro-actively enter into fetch mode and aggressively send out NACKs for the second half of the message. The overall impact of this attack is $O(cn)$ where $c$ is the number of retransmission of NACKs and $n$ is either the total number of nodes in the WSN for broadcasted messages, or the length of the message path, if and underlying routing protocol is used.
1.3.2 DTC

DTC (Distributed TCP Caching) [41] is a specifically modified TCP protocol for WSNs. It provides both up and downstream reliability with hop-by-hop recovery.

**Protocol overview:** This protocol implements a special SACK-based algorithm that can recover missing fragments along the path between the source and the destination. A SACK packet has two fields:

- The *ACK field* contains the sequence number of the last fragment that was received in-order – without gaps. This is similar to the ACK used in TCP protocols.
- The *SACK field* lists the sequence numbers of additional fragments received out-of-order.

It is important to note, that the SACK field also works as a multiple NACK, since it implicitly lists all missing fragments.

DTC assumes, that each intermediate node between an $S$ source and a $D$ destination node can store only one fragment. Periodically, $D$ sends a SACK packet to $S$. Along the path to $S$ each $I$ intermediate node examines the SACK packet. If it acknowledges a fragment that is stored by $I$, the node deletes that fragment from its cache. If the SACK negatively acknowledges a fragment that is stored by $I$, the node retransmits the missing fragment and injects its sequence number into the SACK field. Finally $I$ forwards the SACK packet to the direction of $S$. If an intermediate node can retransmit all missing fragments listed in a SACK, it drops the SACK.

**Weaknesses:** As opposed to NACK-based protocols, ACK-based schemes can achieve full reliability without any further extension. Also, if an attacker injects an ACK into a system, it does not generate any additional traffic. However injected ACK packets can be very dangerous. In general, protocols that use ACKs assume, that an arbitrary fragment which was acknowledged explicitly or implicitly, can be deleted from the system as it arrived to its destination. Since an attacker can forge and insert fake ACKs for fragments that are actually lost, he can cause permanent fragment losses.

In DTC, this attack can be realized easily. A SACK lists multiple lost fragments, so an attacker can forge and inject another SACK that acknowledges all missing fragments. With this single packet, he can provoke multiple fragment losses.

Beside the previous vulnerability, energy depleting attacks are also feasible against DTC. We wrote earlier, that the SACK field also functions as a multiple NACK. Since DTC does not limit the loss window and propagates SACK messages, injecting a SACK with a large loss window – like a packet (NACK: 1, SACK: 255) – generates large traffic. If nodes can store only one message, like it is assumed in DTC, the impact of this attack is $O(sl)$, where $s$ is the size of the loss window, which is maximized by the size of the transmitted data, and $l$ is the length of the path between $S$ and $D$.

In addition, the previous two attacks can be easily combined by injecting an inverse SACK packet to the system, that requests the retransmission of every fragment that was actually received by $D$ while acknowledges every lost fragment.

1.3.3 Garuda

Garuda [119] provides a scalable solution for sink to all sensors communication. It is a downstream reliable scheme using single NACKs, and a local recovery scheme realized by special CORE nodes.

**Protocol overview:** The protocol uses a CORE architecture, where every 3rd node is a CORE node, serving as a local and designated loss recovery server. Nodes use implicit multiple NACKs, to recollect missing fragments, where the NACK is the sequence number of the last message ID the node has received thus far.

To protect against lost full messages, Garuda uses a special *Wait-for-First-Packet* (WFP) pulse, which is a small finite series of short duration pulses, where the series repeated periodically. Sensor nodes upon reception of the pulses, also start pulsing. The sink after pulsing for a finite duration
transmits the first fragment. If a node receives the first fragment, it stops pulsing the WFP and broadcasts the first fragment. Therefore, WFP serves as an implicit NACK for the first fragment, while termination of WFP pulsing is an ACK for it. As first messages can store the size of the data that is going to be transferred, reliable transfer of the first fragment can solve the lost last fragment problem.

**Weaknesses:** The major problem with Garuda resides in the unconditional propagation of WFP pulses. If an attacker injects a WFP into the system, every node will immediately rebroadcast it, until an undefined time. Even if nodes stop pulsing after $c$ consecutive WFPs, the impact of this attack is proportional to $O(cn)$, where $n$ is the number of nodes in the network.

1.3.4 RBC

RBC (Reliable Bursty Convergecast) [184] implements a special window-less block acknowledgement scheme, that can be used for hop-by-hop recovery.

**Protocol overview:** In RBC, intermediate nodes cache every fragment they receive. If a fragment is acknowledged, it is deleted from the cache, otherwise repeated $n$ times. RBC implements a special cache queuing model capable of efficiently delivering out-of-order fragments, which is useful for bursty communication. The protocol uses multiple (block) ACKs.

**Weaknesses:** The window-less ACK is only useful to achieve better throughput, it does not effect reliability. Unlike hybrid NACK-ACK schemes – like DTC – in a protocol that uses solely ACKs, the receiver cannot request the retransmission of a fragment and the sender will never repeat an acknowledged fragment. Therefore a false ACK on a lost fragment guarantees fragment loss contrary to other schemes where recovery might be feasible, although it can have a big overhead. Moreover, as RBC supports block acknowledgements, it is possible to acknowledge every fragment stored by a node in one ACK. Upon reception of this packet, the node will completely empty its cache, which can lead to fragment losses with high probability.

1.3.5 DTSN

DTSN (Distributed Transport for Wireless Sensor Networks) [108,131] was developed in the context of the EU FP6 project UbiSec&Sens by project partner INOV, which is also partner in the WSAN4CIP Project. For this reason, DTSN will likely be used in our further developments, and it is particularly important to analyze its dependability properties.

The main characteristics of this protocol are the following:

- The control packets, like ACKs and NACKs, issued by the final data destination are tightly controlled by the sender, which uses piggybacking on data packets as much as possible.

- Caching of data packets at intermediate nodes serves two purposes: a) to minimize end-to-end retransmissions; b) to increase the probability of delivery of data even if not buffered at the sender (i.e. data that demands partial reliability only, such as precision enhancement data or simply non-critical data).

The core of the DTSN specification is a full reliability service, though the original specification also mentions a partial reliability service. Since the latter is based on the former, only the full reliability service will be analyzed in this section.

**Protocol overview:** This service was thought for critical data transfer requiring full end-to-end reliability. Besides providing full reliability, this service tries to minimize energy consumption and increase network life-time. Full reliability is achieved by a Selective Repeat Automatic Repeat Request (ARQ) mechanism that uses both negative acknowledgement (NACK) and positive acknowledgement (ACK) control packets.
In DTSN, a session is a source/destination relationship univocally identified by the tuple (source address, destination address, application identifier, session number). The session is soft-state by nature both at the source and at the destination, being created when the first fragment is processed and terminated upon the expiration of an activity timer (provided that no activity is detected and there are no pending delivery confirmations). The session number is randomly chosen and appended in order to unambiguously distinguish between successive sessions sharing the endpoint addresses and application identifier. Within a session, data fragments are sequentially numbered. The Acknowledgement Window (AW) is defined as the number of packets that the source transmits before generating a confirmation request (Explicit Acknowledgement Request - EAR), and its size depends on the specific scenario.

The control of delivery confirmation is done at the source to allow the definition of the trade-off between overhead and speed of loss recovery to be application-specific. After sending an EAR, the source launches an EAR timer. If the EAR timer expires before an ACK/NACK is received, the source retransmits the EAR packet.

The DTSN algorithm at the destination works as follows. Upon reception of a data fragment with a new session identifier, a new session record is created. If, on the other hand, the session identifier exists but the session number is different from the recorded one, the session record is reset and the new session number replaces the old one. The destination then collects the data fragments that belong to that flow, delivering in-sequence packets to the higher layer protocol. Upon reception of an EAR, the destination sends an ACK or NACK depending on the existence of gaps in the received data fragment stream. Upon the expiration of an activity timer, the session record is deleted and the higher layer protocol is notified in case there are unconfirmed fragments.

Caching at intermediate nodes is the mechanism employed by DTSN to counter the inefficiency associated with end-to-end retransmissions. In DTSN, each node keeps a cache of intercepted fragments, managed according to a suitable replacement policy, such as FIFO. The fragments are stored in cache with probability $p$, and may belong to different sessions whose end-to-end routing path includes the node in question. Each time an intermediate node receives a NACK packet, it analyzes its body and searches for corresponding data fragments that are missing at the destination. In case a missing fragment is detected, the intermediate node retransmits it. It also changes the NACK contents before resending it, adapting its gap list so that the retransmitted fragments are not included. In this way, the source will only have to retransmit those data fragments that were not cached at intermediate nodes, decreasing the average hop length of the paths traversed by retransmitted fragments. Additionally, intermediate nodes eliminate the cache entries that correspond to fragments whose delivery is confirmed by an ACK or NACK.

**Weaknesses:** Similarly to other NACK-based transport protocols, DTSN is vulnerable to energy depleting attacks. An injected fake NACK forces a WSN to unnecessarily replay the fragment therein declared as missing. The only limiting factor is that DTSN limits the size of the loss window to a given multiple of the AW size. These attacks have the impact of $O(s)$, where $s$ is the number of fragments required to transmit a full message.

### 1.3.6 Other protocols

There are additional WSN transport protocols that offer reliable data transport, however they do not use a new technique or a new concept that hasn’t been introduced before, so we only mention them briefly.

- **RMST** [141] can use a hop-by-hop or an end-to-end NACK-based recovery scheme. Despite the purely NACK based solution, the authors does not deal with the lost last fragment or with the lost full message problem, meaning that the protocol cannot be considered as a reliable scheme.

- **STCP** [173] uses a simple NACK based scheme for continuous flows, while an ACK based scheme for event-driven flows. It is important to note, that NACK-based schemes do not suffer from the two loss problems for continuous flows, as there are no first or last fragments. However, the
purely ACK-based scheme is highly vulnerable to replayed ACK packets, as it was described earlier for RBC.

- Wisden [169] uses a simple NACK based scheme with hop-by-hop or with end-to-end recovery. To overcome the main problems of NACK based schemes, the protocol periodically inserts dummy fragments to maintain a continuous message flow. Although these additional fragments provide a solution for the lost last fragment and for the lost full message problems, they can cause a massive communication overhead, especially for networks observing rare events. Also, there is a trade-off between energy consumption and error detection / recovery rate. If these dummy fragments are inserted more frequently, fragment losses can be detected and corrected faster, however the communication overhead increases and nodes can spend less time in sleep mode.

- Flush [75] assumes, that the link layer provides single-hop ACKs. The protocol uses end-to-end multiple NACK based retransmissions, with a limited loss window. Despite the link layer ACKs, the authors assume that end-to-end fragment losses can occur, but it is unclear how, since single-hop ACK-based protocols can be fully reliable, – in non-hostile environments – however they have significant overhead. By all means, flush does not deal with the problems of the NACK based schemes, so that a lost but acknowledged last fragment can cause permanent losses. Finally, the authors refer to an experiment with a 48-hop path and NACKs containing maximum 21 retransmission request. In a network with these parameters, one properly injected NACK can induce minimum $48 \times 22 \times 2 = 2112$ additional fragments (the NACK and all requested fragments should be transmitted over the whole path, each fragment is ACK-ed in the link layer).

- RCRT [118] uses a multiple NACK based end-to-end recovery scheme along with multiple ACKs, hence it inherits every vulnerability of the two methods, without dealing with the problems of NACK-based solutions.

There are further reliable WSN transport protocols like IFRC [127], Tenet [56] and ESRT [135] that deal with reliability, however the used scheme is describe with insufficient details to analysis on them.

1.4 Lessons learned and future work

Both ACK and NACK-based schemes are vulnerable to injected control packets. In general, ACK-based schemes are vulnerable to attacks against reliability, while NACK-based protocols are only vulnerable to energy depleting attacks. For both methods, the multiple version is significantly weaker. Moreover, if a protocol combines ACK and NACK packets – like SACK-based schemes – then it inherits the problems of both methods.

In practice, attacks against reliability are more important than energy depleting attacks, therefore NACK schemes may be preferred to ACK schemes. NACK schemes are also more suitable for multi-hop communication. However, pure NACK-based schemes have two inherent weaknesses, the lost last fragment and the lost full message problem.

It is relatively easy to solve the lost last fragment problem by informing the destination node about the number of fragments in the message at the beginning of the communications (e.g., in the first fragment). For the lost full message problem, we cannot identify a satisfactory solution for the time being. Garuda was the only protocol that tried to tackle the problem, however it led to a serious energy depleting attack. Perhaps this specific problem requires a dedicated ACK-based technique, while NACKs should be used everywhere else in the communication. It is important to note, that these problems only exist for event driven applications. For continuous communications, NACK-based schemes can be directly applied, as there are no first or last fragments.

Without adequate authentication – probably using cryptographic solutions – it seems impossible to fully protect a NACK-based protocol against energy depleting attacks. However, the impact of these attacks can be kept low with some precautions. If multiple NACKs are used, the loss window must be maximized and hop-by-hop or some kind of local retransmissions should be used instead of end-to-end recovery.
It is hard to estimate an impact threshold where these types of energy depleting attacks become dangerous. Ideally, the ratio between the number of injected packets and the number of packets generated due to the injected packets should be constant, however, this objective might be difficult to achieve, if possible at all. On the other hand, if this ratio is proportional to the size of the network, then the attack can definitely be considered to be dangerous.

While the transport protocols analyzed in this section were not designed to resist malicious attacks, and from this point of view, they cannot be blamed to fail in hostile environments, we must emphasize that we are not aware of any reliable WSN transport protocol that is designed with malicious attacks in mind. Cryptographic protection of control packets would help, but it is unclear in which layer it should be used.

Application of cryptographic mechanisms at lower layers (i.e., in a hop-by-hop manner) may leave the protocol vulnerable to attacks mounted by compromised intermediate nodes. In addition, that approach would likely be inefficient. For instance, as we have seen, WSN transport protocols require intermediate nodes on a path to cache data packets until they can be sure that they have been delivered. Hence, control messages must be authenticated such that these intermediate nodes can verify them, meaning that we need to use a broadcast authentication scheme. Such schemes are either computationally expensive (e.g., digital signatures) or their management cost is high [109], and therefore, they should not be used at the network layer to protect each and every packet. Thus, we need to solve the problem at the transport layer, by enhancing transport protocols with their own security mechanisms. Finding a way to do this efficiently is on our agenda for future work.
2 Dependability of multicast dissemination

For the administration of a WSN island, a dissemination protocol based on broadcast is a useful support service to communicate data to all the nodes, or at least a large subset of them. The payload of the update to be disseminated plays an important role on the requirements of the dissemination protocols. Therefore, we categorize the updates that the sink broadcasts into different size categories and show their related use cases:

- **Small source data size**: We understand by small an update that is smaller than the packet payload size, and thus can be completely transmitted with a single packet. The sink generally issues such an update to set the state of the network service, or to configure a particular parameter. For example, such an update could be used by the administrator to set the level of service of the network.

- **Large source data size**: The main application for a large update in sensor networks is the code update procedure. By large update, we understand a number of packets that is larger than a node can actually buffer in its RAM. At this size, network coding techniques can yield promising performance improvement.

- **Medium source data size**: Finally, there are updates that are just a few packets large. In the current state, they lack a good support tool to be disseminated. For example, they are needed for small code updates, such as applying a patch, or disseminating a revocation list of certificates.

In the WSAN4CIP project, we will investigate the large updates in more details, as we believe there is a prevalent need for such a support mechanism in WSN and that it was not fully addressed in terms of dependability until now. However, we do not underestimate the needs to support smaller update mechanisms, are they are necessary in the daily administration of a WSN.

The design of a large data dissemination protocol shall meet a few criteria:

- **Security**: As efficient dissemination protocols rely extensively on broadcast and multicast transmissions, symmetric approaches offer poor security as the leaking of the key by one node can corrupt or break the whole network security. Asymmetric approaches, possibly combined with efficient symmetric schemes are therefore needed in order to prevent malicious nodes to forge packets.

Confidentiality of the data is also of importance due to the sensitive nature of the information transmitted. For example, if a code image is disseminated, the attacker could reverse engineer the eavesdropped data, and exploit bugs found in the code to perpetrate a remote code injection [52].

- **Reliability**: The update must reach all the destinations, as long as the network is connected, and that there are no on-going DoS attacks. The dissemination protocol should ensure that none of the nodes is left apart. In WSN, we expect high packets losses, and therefore novel transmission techniques such as network coding are studied.

- **Congestion control**: The dissemination of large data files generates unusual traffic abundance in a WSN. This uncommon traffic pattern is not well supported by the MAC layer, whose design aim is generally not the throughput, but energy efficiency and latency. Proper care must be taken in order to avoid superfluous loss due to collisions of simultaneous transmissions.

- **Energy Efficiency**: Sensors are restricted devices, often relying on an autonomous source of energy. Therefore, dissemination protocols and the algorithms they are build upon must remain lightweight and minimize overhead as much as possible.

In the remaining of this section, we first outline the state of the art for disseminating large files in a WSN. We then introduce two key building blocks of our dissemination solution: rateless erasure codes (also known as fountain codes) and fuzzy control. Finally, we sketch out the basic design of our dissemination protocol.
2.1 State of the art

As the main application for large data dissemination for WSN is the OTA code update, there is a lot of papers on code updates in the current state of the art. We of course focus on the networking properties of those previous published protocols, and not on code update specificities.

The most straightforward approach for providing multi-hop dissemination is to simply flood the file to every node in the network. However, flooding, which essentially is na"ive retransmission of broadcasts, can lead to the so-called broadcast storm problem, i.e. collisions contention, and redundancy severely impair performance and reliability [155]. Therefore, several more sophisticated dissemination protocols have been recently developed that aim at an efficient propagation of large data files in ad hoc wireless sensor networks. Among them, XNP was one of the first network reprogramming protocols [68]. It allowed only for single-hop code distribution. Nevertheless, it was included in previous versions of tinyOS.

The Multi-hop Over the Air Reprogramming (MOAP) protocol successfully extended data dissemination to multi-hop networks [143]. It uses a ripple-like propagation mechanism to propagate new code images to the entire network. In MOAP, receivers apply a sliding window to identify lost segments, which are then re-requested using unicast messages to prevent duplication. Eventually, broadcast requests are substituted for unanswered unicast requests. MOAP does not allow for spatial multiplexing, i.e. before nodes can become senders, they first need to receive the complete code image.

Trickle is an epidemic routing protocol that is based on a polite gossiping policy [91]. It builds upon the suppression mechanisms proposed by the Scalable Reliable Multicast (SRM) method a multicast approach for wired networks that controls network congestion and request implosion at the server by applying communication suppression techniques [73]. In Trickle, new data is advertised periodically in the form of small code summaries. On receiving an unknown code summary, nodes eventually request the missing data. Trickle borrows the idea of a three-phase handshaking protocol (advertisement-request-data) from SPIN, a negotiation-based epidemic algorithm for broadcast networks [82].

Deluge is an epidemic reprogramming protocol that builds off Trickle by adding support for large object sizes [64]. Importantly, Deluge introduced the concept of spatial multiplexing, which allows for parallel data transfer. This so-called pipelining offered significantly faster performance compared to previous data dissemination protocols. Deluge is included in recent TinyOS distributions. It is generally accepted as the state of the art for code image dissemination in the field of wireless sensor networks [65].

The Multi-hop Network Programming (MNP) protocol shares many central ideas with Deluge, including the three-phase handshaking protocol, pipelining, and using a bit-vector to keep track on missing packets [83]. In addition, it provides a greedy sender selection scheme that attempts to ensure that only one node transmits data at any one time. In contrast to Deluge, nodes are allowed to turn off their radios while neighboring nodes transmit data that is not relevant to them. Similarly, Freshet’s design aims at improving Deluge’s energy efficiency [78]. Based on some metadata, receiving nodes try to estimate the time until the code image data will reach their vicinity and enter a more energy-efficient stand-by-mode accordingly.

By introducing Sluice, Lanigan et al. contributed the first attempt to secure the code image dissemination with Deluge [86]. In Sluice, authentication of individual pages is based on a hash-chain. However, in using Sluice it is not possible to identify individual forged packets, i.e. a failed attempt to verify a whole page necessitates its complete retransmission.

Subsequently, Deng et al. proposed a hybrid approach “combining hash trees with hash chains” to allow for out-of-order packet verification. In detail, for each page a hash tree is constructed over the individual packets in that page. Each root packet is then concatenated with the hash value of the previous page. Finally, the concatenations are used to create a hash chain whose first element is signed with a public key signature scheme. Lui et al. contributed Seluge, which improved the approach by Deng et al. by using a hash tree of hash chains instead [65]. In addition, Seluge integrates protection against Denial-of-Service attacks.

Recently, Rossi et al. proposed Synapse, a data dissemination protocol that applies rateless LT codes [134]. Synapse showed improved efficiency over Deluge in a single-hop scenario. It applies a Gaussian elimination mechanism for decoding. Data blocks are sent during so-called dissemination
rounds and only one node at a time is allowed to send. Similar to MOAP, Synapse implements only a hop-by-hop data dissemination protocol.

AdapCode also applies network coding and Gaussian elimination [63]. In contrast to Synapse, though, encoding is not based on the computationally cheap exclusive or operation, but on linear combinations of source packets. The coding scheme is adopted dynamically in accordance with the node density. Senders are determined by a distributed selection mechanism, i.e. every potential sender waits for a random period of time and starts transmitting only if it does not overhear encoded packets from another node.

Hagedorn et al. devised Rateless Deluge, an extension of Deluge using random linear codes. To solve the set of linear equations and retrieve the source data, Rateless Deluge applies Gaussian elimination with back substitution [60]. On average it took 6.96 (1.96) seconds to decode a 48 (24) packet page on Tmote Sky motes. The long decoding times were ascribed to the asymptotic runtime complexity, which is for decoding a time matrix. Importantly, neither Synapse, nor AdapCode, nor Rateless Deluge provides any security mechanisms.

Related work on secure fountain codes – Solutions for the poisoning attack of Fountain Codes involve public-key primitives such as homomorphic hash functions or homomorphic signatures [24,79]. However, both methods are not applicable on a per packet basis in a wireless sensor network due to the high data overhead compared to the very short packages and the high computational complexity of public-key cryptography. Another solution using error correction is provided in [69], however this can not offer the high security against the full adversary needed for code image protection.

Related work on fuzzy control for WSNs – The usage of fuzzy logic as well as fuzzy control has recently become popular also in WSNs. The examples are manifold: The work in [59] introduced a fuzzy based approach for the problem of cluster-head election. The fuzzy controller is performed by a central control algorithm run at the base station, which is assumed to have global knowledge over the network. In [87], a fuzzy based mechanism to face selective forwarding attacks - malicious nodes dropping sensitive packets - in WSN. With the protocol FAIR [30] for fuzzy-based resilient data aggregation in WSN some of the authors of this work make use of distributed fuzzy control systems to provide robustness and quality of information in the presence of bogus aggregator nodes. Due to its proven real-time responsiveness distributed fuzzy control is the mean we have chosen also in this work to handle the multi-hop propagation in an efficient and robust way.

2.2 Fountain Codes

For the efficient and robust propagation, one can use rateless erasure codes, or Fountain Codes [105]. The idea of a Fountain Code is that a sender is able to generate a potentially unlimited sequence of code words while allowing the receiver to decode the source data from any subset of encoded packets equal or slightly larger than the number of source packets. Fountain Codes are beneficial since they do not require the receivers to inform the sender about individual missing packets. If packets get lost due to the unreliable medium, it is not required at the sender side to resubmit exactly the missing packet. Instead, due to the fountain characteristic, any new random linear combination $X_{j+k}$ can be transmitted by the sender. Taking that into account and assuming that all parties finally would like to receive the same data stream, fountain codes allow to use the wireless broadcast channel more efficiently. The same encoded packet may, based on their potential different pre-existing knowledge about previously successfully received encoded packets, allow different receivers to filter out different information relevant to them.

The first efficient realization of this idea are LT Codes [103]. An encoded packet is computed in two steps assuming that preliminarily a data block is separated into $m$ packets:

1. A packet degree $d$ is randomly chosen according to a given distribution. The choice of the weight distribution is the key parameter with respect to the performance and the efficiency of the coding scheme $\rho(d)$.

2. The encoded packet is obtained by choosing uniformly randomly $d$ out of the $m$ source packets,
namely \( \{p_{\ell_1}, ..., p_{\ell_d}\} \), and successively XORing them to compute

\[
X = \bigoplus_{i=1}^{d} p_{\ell_i},
\]

This is done for at least \( N > m \) encoded packets \( X \). The information which packets \( p_{\ell_i} \) have been considered for a concrete encoded packet \( X_j \) is represented in a coding vector \( C_j \) of size \( 1 \times m \). We denote the degree of a coefficient vector \( C \) with \( D(C) \) and its \( i \)-th bit with \( C[i] \).

The receiver’s decoding procedure is equivalent to solving the linear equation system \( t = Gs \) for \( s \). The \( m \times m \) matrix \( G \) consists of \( m \) linear independent coefficient vectors of successfully received packets, whereas the vector \( t \) contains the corresponding incoming encoded packets \( X \). The vector \( s \) is the vector of all \( m \) plaintext packets which shall be computed. While solving the linear equation system is the optimal decoding algorithm, it is not very efficient requiring in the standard way roughly \( m^3 \) computation steps. The decoding effort can be reduced by choosing a suboptimal decoding process, the so called LT decoding process, and a degree distribution \( \rho(d) \) adapted to this decoder.

**LT Decoding Process:** The LT decoding process is depicted in Figure 2. It uses a buffer \( A \) where not yet decoded packets are stored and a buffer \( B \) for decoded information. Encoded packets \( X, C \) received over the radio interface are decoded in accordance to the information in the coding vector \( C \). Those plaintext packets \( p_i \) which are already stored in the buffer \( B \), and which are relevant according to the currently processed packet \( (X, C) \), will be applied to the actual decoding. The processing is illustrated in the Algorithm 1 Decode.

**Algorithm 1 Decode**

| Require: \( X, C, p_i \in B \) |
| Ensure: \( X', C' \) |
| 1: for all \( p_i \) in \( B \) do |
| 2: if \( C[i] = 1 \) then |
| 3: \( X' = X \oplus p_i \) |
| 4: \( C'[i] = 0 \) |
| 5: end if |
| 6: end for |

If after passing the algorithm decode the remaining encoded packet \( (X', C') \) is of degree \( D(C') = 1 \), i.e. it is actually a source packet \( p \), such that it is added to the list of decoded packets in buffer \( B \). If the degree remains larger than one \( (D(C') > 1) \), it is inserted in the buffer \( A \).

The buffer \( B \) stores all the actually decoded plaintext packets \( p_i \). If a new element is added to \( B \), all \( (X, C) \) in buffer \( A \) will again be applied to the decoding process.

Conceptually, the buffer \( A \) is unlimited in size which obviously will not be the case in a real-world deployment on restricted devices. Here, a good design choice is to limit the buffer size of \( A \) and \( B \) to \( A + B \leq n \) with \( n \) denoting the number of cleartext packets belonging to the page \( P \).

Since the decoding is mainly based on XOR operations, the LT decoding process is extremely efficient. However, the limiting factors on a sensor node are the data overhead and the buffer sizes for buffers \( A \) and \( B \).

The efficiency of LT Codes depends largely on the degree distribution of the encoded packets. Due to the decoding, a high number of packets with low weight need to be present. I.e. the decoding process cannot start before a packet with weight 1 is received. On the other hand, the redundancy should be minimized such that a set of slightly more that \( m \) packets contains the full information.

### 2.2.1 Confidentiality of Fountain Codes

Network coding techniques such as Fountain Codes are a promising way to propagate large bulks of data in a multicast manner over an unreliable medium. However, there may also the need to conceal
such encoded data streams on its way to the receivers. Compared to conventional 'encrypt - encode / decode - decrypt' approaches, other solutions may be preferable due to two reasons: i) they cause to orders of magnitude less CPU investment for encryption and decryption; ii) besides hiding the data, one may also want to hide the coding information from an eavesdropper.

The network coding paradigm has recently been applied to WSN code image update scenarios. Obviously, once deployed there is a growing need to enrich such concepts with security means. Solutions regarding the integrity and authenticity of incoming encoded packets have already been proposed in [19], and [67]. On the other hand there is currently no work which explicitly deals with the confidentiality of encoded data. This topic may be of significant importance when transmitting a code image over the wireless. Otherwise code analysis may be a very good starting point for attackers to find weaknesses in the implementation of sensor nodes.

This may have two reasons: one camp argues that the encoding in itself is already a weak mean of encryption and therefore additional protection is not required anymore. Examples following this direction are [19], and [150] assuming that the attacker can eavesdrop only on a subset of all transmission paths between source and destination(s). The other camp argues that there is no research challenge in weaving confidentiality into the network coding paradigm by applying an encryption transformation $E$: the only decision to be taken is whether to encrypt on the plaintext data or to encrypt on the encoded data. This camp further argues to transmit either $E(X_i, C_i)$ or $E(X_i) , C_i$, or to apply the encryption on the plaintexts yet $E(p_i)$ before generating the $X_i$. We state that all these approaches have their drawbacks either with respect to security, CPU investment or with respect to the flexibility for generating new encoded packets on its way to the final multicast destinations.

Our contribution in WSAN4CIP is to conceal the data stream of network encoded packets by at the same time allowing intermediate nodes to generate new encoded packets based on already received ones. This will be done in a lightweight manner by also hiding information about the composition of the encoded packets. One example where we see value for our solution are environments with restricted devices and/or in cases where energy saving of security enabled devices due to eco-IT aspects is an issue. More details regarding our solution will be given in Deliverable D3.2.

### 2.2.2 Network Coding and dependability challenges

While in the single-hop data stream transmission scenario, all packets are encoded at the base station, in a multi-hop scenario, intermediate sensor nodes may need to encode packets by themselves. Pure forwarding of encodings from the base station would use the benefits of Fountain Codes only at the first hop. Encoding of information in a node inside the WSN is a form of network coding\(^1\). Network Coding, however, causes new threats by poisoning attacks and requires new security solutions to ensure a dependable service also under such circumstances.

An attack to network coding systems is the so-called pollution attack, a special form of a denial of service attack where modification of a single bogus packet may affect several source packets. This

\(^1\)Different to common practice in network coding, this is not done by combining several encoded packets, but by decoding and re-encoding.
is caused by the transmission of coded packets that for themselves do not carry usable data. Only if enough well-formed packets are received, the decoding process to recover the data stream can be started. By modifying only a few blocks, an adversary has the power to prevent the receiver from decoding any block. Even worse, if the receiver is a node that performs network coding, i.e. combining the malicious packet with well-formed packets, the node produces new malicious packets that spreads the error throughout the network. In the end, this may prevent successful decoding for all receivers.

2.3 Our approach on dependable dissemination for WSNs

We derive now the fundamental pillars that differentiate our desired solution in regards to the prior work listed in section 2.1.

- **Fountain codes:** Two properties dictate the choice of fountain codes for the network coding. First, they perform well in “lossy” networks, as they are erasure codes, and WSN are often considered as lossy. Furthermore, they perform well in dense networks, where the different receivers might drops different packets, but thanks to the fountain coding, this has no impact on the sender. It just has to send more fountain encoded packets, with little care of what exact source packets the receivers are missing.

- **Sender selection:** Most of the existing solutions use a sender selection process in order to reduce packet collisions on the wireless medium. Basically, the desired output of a sender selection is to pick up only one forwarder in a one-hop neighborhood. This reduces the complexity of the dissemination protocol, where only the few selected senders need to keep a state and possible re-transmit loss packets.

  However, the stateless aspect of the fountain code offers an interesting opportunity: To transmit a specific chunk of data, multiple senders do not need to coordinate to transmit it. They all pick up independently random combinations of packets, and send the encoded result. This has the advantage to use the spatial diversity of the network and to spread losses more equally among nodes, but it also increases the number of collisions.

- **Expedite the dissemination:** With current technologies, the power needed to listen on the radio and receive packets is equivalent to the power needed to send [1]. To deal with these high receiving energy costs, we propose to expedite the transmission of the file as fast as possible, thus having no sleep cycle during the dissemination of large files. Once the dissemination completed, nodes can return to a normal sleep cycle.

  In the LPL MAC layer of tinyOS 2.0, this is done naturally. A node remains awake for a short amount of time after receiving a packet, in order to receive bursts of packets efficiently. In high throughput scenarios, the node remains awake continuously.

- **Packets collision prevention:** To diminish the number of collisions, while keeping a fast dissemination time, we propose to use fuzzy control that takes into account dissemination statistics (number of NACKs, number of packets overhead), as well as counting number of corrupted packets received. Thus, the fuzzy control can throttle the sending rate and other parameters of our dissemination protocol, in order to keep the dissemination performance optimal.

- **Packet-wise authentication:** Many dissemination protocols only rely on link layer authentication, or on authenticating the full image only. This leaves a breach open for the attacker to implement DoS attacks at a very little cost. We propose to protect every packet at the application layer, in order to mitigate pollution attacks.

A complete protocol specification is to be reported in Deliverable D3.2.
3  Dependability of routing protocols

There is a vast literature on wireless sensor network routing protocols. The large variety of routing protocols is due to the diverse application requirements and network assumptions. Routing surveys that have been proposed so far attempt to make an exhaustive list of existing routing protocols and/or classify them based on some rough network and operational characteristics. However, that approach has the following problems:

- First, that approach hardly supports the development of sensor network applications due to the rough operational and network models. In particular, using those classifications, it is difficult to identify a routing protocol which perfectly fits specific application requirements. The designer has two choices: he can either select a routing protocol that partially satisfies the application requirements and provides a “good enough but not perfect” solution, or he must develop a novel protocol for the specific application using a clean-slate approach. The first choice results in a suboptimal solution, whereas the second approach requires lot of effort and results in a protocol that lacks intensive analysis.

- Second, existing routing surveys tend to disregard proposals that are not concerned with full routing protocols but only deal with protocol components. Prior works on sensor network routing are diverse which means that the proposed routing components are often independent and can be jointly used. For instance, some works focus on path selection, while others deal with different cost metrics and their calculation.

- Third, existing surveys do not consider the dependability attributes of sensor network routing protocols. For instance, there are separate surveys on the security of WSN networking protocols, and routing surveys hardly contain any secure routing protocols. This approach promotes the two-step development of routing protocols where they are made dependable only after their initial design and implementation. However, dependability is a part of routing objectives, and as such, it should be considered from the grounds as a basic design principle. Moreover, dependability also includes reliability and maintenance attributes besides security which are not considered by any routing surveys.

Instead of creating yet another survey of routing protocols, here we attempt to factor out the main design principles for sensor network routing, as well as to identify the most important dependability concepts in this context.

We imagine that a routing protocol is a combination of different routing modules where each module has one or more routing objectives (like real-time or dependable packet delivery), and a designer should select among these modules based on the required objectives and a given routing model. This modular approach lies between the two extremes detailed above. Each module may have different implementations in different works, where each work has different routing model (i.e., network and operation assumptions). Therefore, we identify the mainstream implementations of different modules, and give their routing model. We emphasize that this list of implementations is not intended to be exhaustive, it rather serves as a starting point as well as a demonstration purpose for our methodology.

In addition, in contrast to prior works, we also classify all modules (and indirectly routing protocols) according to their dependability attributes (like availability, reliability, security and maintainability) that enables designers to consider dependability objectives as a basic design principle.

The organization of this section of the report is the following: Subsection 3.1 gives an overview of our modular approach to the design and analysis of WSN routing protocols. In Subsection 3.2, we describe the network and operational models of WSNs. In Subsection 3.3, we introduce various routing objectives. Finally, in Subsection 3.4, we describe the modules and the components of routing protocols that can be used to achieve the different routing objectives, and in Subsection 3.5, we summarize our approach and give an outlook to future work.
3.1 Overview of our proposed modular approach

Our approach builds upon the network and operational model, a set of routing modules, and the routing objectives. Instead of selecting a specific protocol, an application designer should identify routing modules which try to achieve the desired routing objectives. The routing objectives define the goals of all routing modules like the guarantees of packet delivery with real-time constraints and dependable requirements. Afterwards, an implementation of the module can be chosen which matches the network and operational model of the application. All modules are categorized into four different components.

The low-layer component includes all modules which directly invokes the data-link layer in order to conserve energy as well as to increase reliability and network throughput. In particular, these modules can measure link reliability to aid routing decisions, use network coding or error-correction to reduce retransmissions, or implement reliable broadcasts by exploiting node overhearing. These modules provide different link-layer measurements and/or topological information to upper-layer modules.

The cost calculation component encompasses all routing cost calculation modules. These modules may need some input from the low-layer modules such as reliability or power transmission measurements and assign a cost value to a node in the network. This cost value may incorporate energy-based, distance-based, link-reliability based, time-based, or maintenance cost based metrics. This is a core component which means that a routing protocol must include at least one cost calculation module.

The path selection component selects a path towards a destination based on the available routing information delivered by low-layer and cost-calculation modules. This component includes modules which implement a mean of path selection like centralized selection when a single node computes the routing tables of all other nodes in the network, multi-path selection, probabilistic selection, or route selection towards multiple base stations. This is also a core component (i.e., a routing protocol must include at least one path selection module).

Finally, the security component gathers all modules with specific security goals like data authentication and confidentiality, or misbehaving detection. These security functionalities may be invoked by all modules in all components.

The relation of all modules and components are depicted in Figure 3.

Figure 3: A modular view on WSN routing protocols. Darker boxes denote core components.
3.2 Network and operational model

3.2.1 Network model

**Base station.** It is commonly agreed that the base station is a powerful device with unconstrained energy supply and computational capacity. However, the following characteristics of a base station may severely influence the operation of a routing protocol.

**Number:** In most practical applications, the increased number of base stations provides more robust data gathering, and may also decrease the network delay. However, the typical number of the base stations is one. If only one base station is presented (and there is no need for explicit communication between sensor nodes), the destination node for all messages is identical, while in case of multiple base stations, the destination node may vary.

**Mobility:** In some applications, where the number of base stations is too small to ensure acceptable network delay and robustness, the base station supports mobility during data gathering. This property of the base station severely affects routing, since some nodes in the network field cannot follow the movement of the base station and are not aware of its current position. Hence, the routing mechanism needs to find the mobile base station in the field. Moreover, the routing topology may heavily vary in time that causes extra overhead in the network layer. A few routing modules support mobile base stations, while others tolerate limited mobility.

**Presence:** The base station can be either continuously or partially presented during the routing process. In the latter case, the routing protocol must support the temporary lack of a base station (e.g., the base station is switched off for a certain amount of time due to maintenance reasons), since a missing base station cannot definitely mean a failure. Thus, the messages should not be dropped or rerouted rather their delivery should be delayed.

**Sensor nodes.** In most sensor networks, sensor nodes are homogeneous tiny devices with constrained energy supply and computational capabilities. In addition, we assume that all sensor nodes are stationary. The following characteristics of sensor nodes may differ for some networks, and they can influence the protocol operation.

**Deployment:** Sensor nodes can be deployed in either a deterministic or a random fashion. When nodes are deployed along a road-side, or in a metro-station, the deployment is rather deterministic than random. In these cases, the protocol should adapt to the fixed network topology. However, numerous routing protocols proposed so far rely on the more general random deployment (e.g., nodes are scattered from a helicopter).

**Addressing:** The task of routing in sensor networks is to deliver the queries to the sensor nodes which have the requested data (in case of query-driven routing protocols, see later), and to return the requested data to the querier node. Accordingly, we can distinguish the addressing method of queries and responses:

- **Query-addressing:** All routing protocols which use query dissemination in the networks employ data-based (What is the average temperature?), or location-based addressing (What is the average temperature in location \((x, y)\)?). Here, the location can also be a virtual location which means that they are calculated based on the connectivity graph of the network instead of exact geographic positions (e.g., all nodes in the network can determine their distances measured in hop-counts from the same pre-defined landmark nodes. Then, these distances for each node constitute a vector that is further used to address the node.)

- **Response-addressing:** The response is either returned on the reversed path which the query traversed, or it is routed back purely based on location information. In the former case, neighboring nodes use locally (or globally) unique identifiers to identify the neighbor from which they received the query, and which is further used to forward the reply towards the destination.
3.2.2 Operational model

Communication pattern: A routing protocol can support the communication from sensor nodes to sensor nodes, from base stations to sensor nodes, as well as from sensor nodes to base stations.

- **Node-to-Node:** Generally, there is no need for this kind of communication in sensor networks. However, in some special applications where it is needed, a few routing modules supports this pattern, or alternatively, ad hoc network routing protocols can be employed.

- **Node-to-Base station:** This pattern is usually supported in order to route responses back to the base station. This is typically reverse-multicast (many-to-one), a.k.a. convergecast, which means that every sensor node is able to send a message to any base station. If there are multiple base stations or only one node is responsible for gathering and transmitting the sensed data to the base station, this pattern can also be unicast.

- **Base station-to-Node:** This is the pattern of routing requests originated from the base station to sensor nodes. This is typically anycast (one-to-many), which means that any sensor node which has the requested data can respond to the query. If some nodes are uniquely identified in the network (by their IDs, locations, etc.), then multicast (one-to-many) and unicast (one-to-one) patterns can also be supported. The base station(s) must be capable of sending messages to any sensor nodes.

Reporting model: The reporting model describes what initiates data reporting. In this sense, we distinguish time-driven, query-driven, and event-driven protocols.

- **Time-driven:** Employing a time-driven routing protocol, a sensor node is triggered in specific moments, when it should perform its measurement task, and forwards the measurement to its next-hop neighbor. These activations can be periodic or one-shot in time. Short periods may cause more traffic in the network, and the quality of routing in terms of energy efficiency becomes a crucial concern. Time-driven sensors may be pre-programmed, or the reporting schedule may come with explicit queries. Furthermore, a time-driven routing protocol can support in-network processing (like data aggregation) on intermediate nodes.

- **Query-driven:** The task of a query-driven protocol is to route the queries to the measurement area, and to route back the response to this query. A query-driven routing protocol can also support data aggregation on intermediate forwarders.

- **Event-driven:** A sensor node sends a measurement towards the base station only if a given event occurs (e.g., the temperature falls below a certain threshold). An event-driven routing protocol can support data aggregation on intermediate nodes.

3.3 Routing objectives

Some sensor applications only require the successful delivery of messages between a source and a destination. However, there are applications that need even more assurances. These are the real-time and dependability requirements of packet delivery.

Real-time delivery: The assurance of message delivery is indispensable for all routing protocols. This means that the protocol should always find the route between the communicating nodes, if it really exists. This correctness property can be proven in a formal way, while the average-case performance can be evaluated by measuring the message delivery ratio.

Additionally, some real-time applications require that a message must be delivered within a specified time, otherwise the message becomes useless or its information content is decreasing after a time bound. Therefore, the main objective of these modules is to control the network delay. The average-case performance is evaluated by measuring the message delivery ratio with time constraints.
**Dependable delivery:** In general, dependability encompasses the following attributes: *availability, reliability, safety, security, and maintainability.*

Theoretically, in case of routing, availability means the readiness for correct routing service, where correct routing service is delivered when the service implements the routing function (i.e., it delivers the given packets from the source to the destination). Availability is usually a measure of the delivery of correct routing service with respect to the alternation of correct and incorrect routing service. In general, all techniques which aim at maximizing the network lifetime and increasing the reliability of the routing service belong to this category. Maximization network lifetime is crucial for those networks, where the application must run on sensor nodes as long as possible. The protocols aiming this concern try to balance the energy consumption equally among nodes considering their residual energy levels. However, the metric used to determine the network lifetime is also application dependent. Most protocols assume that every node is equally important and they use the time until the first node dies as a metric, or the average energy consumption of the nodes as another metric. If nodes are not equally important, then the time until the last or high-priority nodes die can be a reasonable metric.

Reliability refers to the continuous delivery of the correct routing service, and it is a measure of the time until a routing failure occurs. These techniques usually achieve reliability by increasing packet delivery ratio. Safety is simply the absence of catastrophic consequences of routing malfunction on the user(s) and the environment, and it is a measure of the time until the occurrence of a catastrophic routing failure. As routing safety is usually considered to be as routing reliability with respect to catastrophic failures, we do not distinguish routing safety and reliability in the sequel.

Note that availability and reliability are strongly related attributes of routing dependability. All mechanisms that increase the reliability of the routing service usually also increase its availability. However, there are some techniques which primarily intend to improve the availability of the service, and not its reliability. These include all mechanisms that attempt to maximize the network lifetime. Clearly, the application of such techniques does not affect the continuity of successful packet delivery, but rather the time how long the service can be eventually invoked.

Security refers to the ability to prevent or mitigate malicious faults that are deliberately caused by the adversary in the routing service. All mechanisms that prevent an adversary to cause malicious faults in the routing service belong to this group. These include all modules which attempt to increase reliability and can be successfully used against some attacks. For instance, multipath routing, blacklisting, route reconfiguration, probabilistic forwarding, link-reliability metrics, and using multiple base stations can mitigate malicious packet dropping, in case message authentication is assumed.

Finally, maintainability refers to the ability to undergo route repairs, and it is a measure of the time of the continuous delivery of incorrect service. Maintainability includes all techniques which help the routing service to recover from faults.

### 3.4 Routing modules

This section details the identified routing modules. Table 1 contains the routing objectives of each module, whereas Table 2 lists the mainstream implementations of each routing module. Note that a routing module can have multiple objectives, and a single work can propose implementations for multiple modules. Finally, in Table 3, we identified the network and operational model of these implementations. In this table, ‘×’ denotes that a feature is not supported at all by an implementation, while ‘*’ means that all values of a feature are supported.

#### 3.4.1 Low-layer modules

Low-layer modules rely on the functionality of the data-link layer to to achieve better performance in terms of network delay and energy consumption.
Cross-layer module: This module is strongly integrated with the data-link layer (as part of a cross-layer design) and exploits the capability of tuning the transmission power of the sensor devices [26], or identifies the best forwarding candidate during a MAC-layer handshaking (e.g., by means of distributed contention [133]). Adjusting the transmission power, every node can calculate what energy level should be used to transmit a message to a neighboring node. This energy level may be inversely proportional to the cost assigned to the neighboring node.

This module helps to achieve higher delivery ratio, which means that this design can also increase the reliability of the routing service.

Cooperative forwarding: Cooperative forwarding exploits the broadcast nature of wireless communication to improve energy efficiency and packet delivery ratio. Nodes buffer packets, and when enough information have been received to recover the original packet, a packet combining procedure is executed. This packet combining technique, which can be based on network coding, or error correcting codes, exploits the broadcast medium and spatial diversity of a multi-hop wireless network by using packets overheard at any node. For example, in [39], nodes combine corrupted packets into correct packets. This protocol allows one node to receive two or more corrupted versions of a packet from its upstream nodes through overhearing, and then recovers the original packet by combining the corrected versions of the packet into the original one. Cooperative forwarding has been shown to increase the delivery ratio [39]. Cooperative forwarding is usually strongly integrated with the data-link layer, and it should disregard the mutable parts of a packet from packet combining (i.e., these parts are modified at each hop). Thus, a minimal interaction with the routing protocol is also needed to detect such packet parts.

Cluster-based (opportunistic) forwarding: Cluster-based forwarding also exploits the broadcast nature of wireless communication to improve energy efficiency. These techniques can be used in conjunction with any routing protocol to achieve better energy-efficiency by reducing retransmissions. The idea is that each node forms a cluster such that any node in the next-hop’s cluster can take forwarding responsibility. This is motivated by the fact that link quality shows significant variability especially in wireless sensor networks, which would normally require several number of retransmissions from the MAC layer in order to successfully deliver a packet. Two subgroups can be further distinguished.

In the first subgroup, two mechanisms are proposed to diminish the number of retransmissions [20, 163]. The first is to use “helper nodes”, which reduces the number of retransmissions by adaptively migrating packet forwarding tasks from weak links to strong links. This means that, instead of retransmitting a packet, the sender “delegates” the retransmission to an intermediate node which has a better quality link to the intended receiver and, opposed to the receiver, has already received the packet by the first transmission. Second, CBF takes advantage of the occasionally successful transmissions over long (and likely lossy) links. In particular, if a (distant) node receives the packet which is closer to the final destination, then the sender does not need to retransmit the packet, because this distant node can forward the packet towards the destination. The module proposed in [20] lies between the data-link and networking layer and it can be used in conjunction with any routing protocols.

Those techniques belong to the second subgroup which also rely on overhearing, and mainly used to implement reliable broadcast protocols. The first time a node hears a broadcast it retransmits the packet unconditionally, as in a normal flood. As additional neighbors transmit the same packet, the node listens and overhears which neighbors have propagated the broadcast. If each node is aware of its one-hop neighborhood, it determines the number of neighbors that are guaranteed to have seen a packet. When this number falls below a predetermined threshold, a node will again retransmit the broadcast packet. This threshold is tuned according to neighborhood density, as higher density neighborhoods require lower thresholds; other neighbors are likely to broadcast as part of the same flood. The protocols belonging to this subgroup (e.g., [142] [74]) can be used with any routing protocols that rely on global broadcast communication.
3.4.2 Cost calculation modules

These modules are responsible for the computation of the routing cost which is used to select the next-hop forwarder (or route) towards the destination.

**Energy-based cost:** The routing cost, which is assigned to next-hop forwarders or routes, can incorporate energy-based metrics in order to prolong network lifetime. These metrics include the residual energy of neighbors to avoid their fast depletion, or the average power level needed to send a packet in order to minimize the energy costs. For instance, in [138], the energy cost of a forwarding candidate is calculated as $e^\alpha \cdot R^\beta$, where $e$ is the energy used to transmit and receive on the link, $R$ is the residual energy of the candidate, and $\alpha, \beta$ are tunable weighting factors.

Energy-based metrics have a strong relation to link reliability based metrics. In particular, several experimental studies on wireless ad-hoc and sensor networks [164, 186] have shown that wireless links can be highly unreliable and exhibit high packet drops. This results in drastic reduction of delivery rate or increased energy wastage if retransmissions are employed. Therefore, combining the expected number of transmission into routing costs [26, 137] results in lower decreased energy costs and higher delivery rate. For instance, modifying the above energy metric accordingly, $e$ can be calculated as $E(p) \cdot R(p)$ [26], where $E(p)$ is the energy level consumed for transmitting a packet at power level $p$, while $R(p)$ is the expected number of transmissions before the sender successfully delivers a packet to the candidate using power level $p$.

**Distance-based cost:** Each node has a position which is used to calculate the distance between any pair of nodes in the network. This distance is either calculated based on the network’s connectivity graph and measured by hop-counts, or it is the Euclidean distance of nodes computed from their geographic positions. In the former case, if a node has a single coordinate (i.e., the number of hops between the source and the destination) an additional unique network identifier of the destination is needed to successfully deliver packets. This metric is employed by the basic version of several routing protocols such as INSENS [37]. In addition, these protocols usually require the discovery of the destination before data forwarding which results in additional costs. In the latter case, each node is aware of its own geographic position, which is used to implement geographic routing. Therefore, unique network identifiers are not needed, as positions are unambiguously assigned to nodes which also eliminates the discovery of the destination in case its position is a priori known. Alternatively, a node can calculate its (virtual) position by measuring its hop-count distance from several pre-defined landmark nodes, and using a similar routing technique like in geographic routing, this virtual position is further used to route data packets towards the destination. Geographic and virtual position based routing is also called as location-based routing protocols.

The advantage of location-based forwarding is that it is scalable (e.g., there is no path setup and recovery latency), it is suitable for both critical aperiodic and periodic packets, and the per-packet path discovery results in self adaptation to network dynamics. In addition, it seems to be more robust against different routing attacks due to its stateless nature (more precisely, routing states consist of the locations of neighboring nodes). On the other hand, each node must be aware of its own position which may require extra hardware components (like GPS), or the extra communication of location coordinates. Moreover, due to its stateless nature, each data packet carries extensive routing information (i.e., node coordinates) which further increases communication overhead.

Geographic positions can be pre-programmed before node deployment or retrieved using external GPS [12, 26, 44, 48, 50, 72, 76, 77, 81, 88–90, 95, 132, 133, 144, 165, 182]. By contrast, virtual positions are obtained by using only connectivity information, and thus, there is no need for GPS-capable devices. The drawback of these solutions is that a position is described by a location vector which typically have more than 2 or 3 coordinates (e.g., in case of BVR [51] this is around 10 in order to ensure acceptable delivery ratio) which causes extra communication costs as each data packet must carry at least the location of the destination [14, 21, 46, 51, 115, 153, 154, 187].
**Content-based cost:** Most sensor applications are data-centric, which means that it is more important what data is asked for rather than who the originator is. In particular, using content-based forwarding, a query is addressed by the data itself (like what the average temperature is or whether there is an alarm situation) and not with the sensor’s address. The base station subscribes to interested events by sending queries which specifies the interested data (this also can be a complex query), and a sensor node which can resolve the query sends a response back to the base station. In the simplest case, a query floods the entire network, but next-hops can be selected by using more sophisticated information theoretic metrics.

**Link reliability based cost:** The routing cost can incorporate some link-reliability metric. For instance, this can be a slightly modified version of the expected number of transmissions (ETX) which considers forward and backward reliability to identify high throughput paths [164]. Such a metric allows the routing protocol to consider cumulative link reliability over paths, and find the most reliable end-to-end path. As link delivery rate changes over time due to environment or transient traffic characteristics and link statistics needs to be reasonably responsive to these changes, the estimation of link quality is required [164]. There are active or passive techniques to collect link statistics. Active techniques rely on periodic broadcasts containing link statistics about each neighbor. This can incur higher control message overhead if link reliability changes frequently. Passive probing involves piggybacking link statistics to the outgoing data packets.

**Time-based cost:** This category includes all metrics which incorporate the propagation delay of routing messages and are used to select a path which satisfies certain real-time conditions. In [117], the propagation delay of control messages are taken as a selection criteria, and thus, it attempts to select the quickest path between the source and destination. In [44], a network wide speed of packet delivery for real-time guarantee is ensured. Particularly, each node maintains the average delay to each neighbor and uses this to evaluate the packet progress speed of each neighbor node and forwards a packet to a node whose progress speed is higher than a pre-specified lower-bound speed $t$. If each node can find a neighbor that can progress a packet with a speed higher than $t$, $t$ can be guaranteed in the whole network. A similar approach is employed in [26], where each data packet carries a time-stamp that is used to calculate the required speed $v$ of the packet at each hop. Those neighbors are considered as potential forwarders, which can provide higher reception speed than $v$. The delay on each link is estimated as the function of the transmission time of the packet, the contention delay (the time needed to acquire the channel), and the expected number of transmissions before the sender successfully delivers the packet.

**Maintenance based cost:** In case some nodes become out of order (e.g., they run out of their energy supply), they are needed to be repaired or replaced. The frequency and the cost of these maintenance activities highly influence the time needed to recover the routing service, and eventually the maintainability of the routing service.

The frequency and the cost of maintenance operations in a sensor field is essentially dependent on the way nodes are depleted. As routing protocols mainly influence the energy consumption of sensor nodes, they can help to create a favorable depletion profile which considers maintenance efficiency. For example, if some nodes are deployed on the top of some trees, while others are not, the maintenance cost of the nodes on the trees are likely to be considerably higher. Thus, a maintenance cost aware routing protocol should carefully use these nodes to forward data.

Note that this metric, which is first proposed in [8], can be combined with most routing protocols by simply incorporating the maintenance cost into the routing cost metric.

If a node stores only negligible amount of routing information like the positions of neighbors or its own routing cost, the module is stateless. Otherwise, when a node may need more extensive processing or storage resources, the module is stateful. Note that most routing protocols combines multiple metrics into a single routing cost. For instance, in [50, 88, 133, 165], the geographic distance is combined with link reliability based and energy-based metrics, while in [26], a time-based metric is also included.
3.4.3 Route selection modules

These modules are responsible for the selection of a route towards the destination.

**Probabilistic selection:** The next forwarder is selected probabilistically, where higher probability is assigned to low-cost routes or forwarders. For instance, in [138], the forwarding probability between nodes $i$ and $j$ is calculated as $p_{i,j} = \frac{1}{C_{i,j}} \prod_{k \in N_i} \frac{1}{C_{i,k}}$ in a decentralized manner, where $C_{i,j}$ is the cost between nodes $i$ and $j$, and $k$ is the index of $i$’s neighbors.

Probabilistic forwarding aids load-balancing, achieves route diversity, and thus, increases routing reliability.

**Hierarchical selection:** Employing hierarchical routing protocols, a hierarchy level is assigned to each node, and a node only forwards those messages that are originated from a lower-level node. This also helps in-network processing, as a node can aggregate incoming data before forwarding that to upper-layer nodes. The base station resides on the top of the hierarchy. The hierarchy construction can be dynamic or static. Using dynamic construction, the role of the cluster head (CH) is rotated, and all nodes belonging to the same cluster will forward all data to their elected CH. The aim of forming this hierarchy is to prolong the network lifetime and to increase reliability.

**Late selection (broadcast-based forwarding):** Each node blindly rebroadcasts all received data packets, and each receiver decides whether the received packet should be rebroadcast or not. The decision can be based upon who sends the message, who the originator is, who it is destined to, or what state it has (e.g., accumulated routing cost). Therefore, broadcast-based forwarding is simply the passing of routing decisions to the next-hops. This technique may increase the robustness of delivery, as all neighbors receive the data packet and can easily take over the forwarding responsibility of neighboring nodes. On the other hand, it can have significant communication and storage overhead.

**Centralized selection:** Each sensor node selects the next-hop towards the destination either by itself using locally available routing information exclusively in a decentralized manner, or every node sends its neighbor list (and the corresponding routing information) to the base station which then computes the next-hop forwarders for all nodes in the network in a centralized manner. Although centralized computation gives optimal solution, it may yield heavy network communication and it is not scalable.

**Route selection towards multiple base stations:** In order to improve the robustness of data collection, multiple base stations (or drains) may be employed. The aim of using multiple base stations is two-fold. First, if the size of a sensor network grows, the paths between the base station and sensors become longer. Thus, the energy consumed by each node to route data to the base station will increase, thereby reducing the lifetime of the nodes. The energy consumed in forwarding the data may be reduced if multiple base stations are employed. This can be implemented by requiring each node to route data towards either a single base station, or to multiple base stations using multi-path routing. Second, in order to be resilient to any single base stations failures, every sensor is required to route data towards two or more distinct base stations. Therefore, employing multiple base stations increases the reliability of the routing service.

**Multi-path selection:** Multipath routing, which encompasses delivering of data packets on multiple paths towards the destination, is a common technique to achieve robustness and load-balancing. The multiple paths between the source and the destination can be partially or completely disjoint and they are maintained at the expense of increased energy consumption and traffic generation. Apart from load-balancing and robustness against node failures, multi-path routing also inherently provides some defense against malicious packet dropping; in order to prevent a packet to reach the base station, the adversary must control a node on each used path to drop the packet. Multi-path techniques used in sensor networks can be divided into three groups:
• The source makes multiple copies of a packet, and routes these copies on different paths in order to increase robustness [66] [53]. These paths can be calculated in advance and maintained proactively by sending data packets at a low rate only on these paths [53]. Alternatively, if the sources have data to send, they flood the whole network with data packets at a low rate, and the destination selects the best quality paths according to some network metric [66]. In [53], two further localized methods were proposed to build multiple disjoint paths and braided (partly disjoint) multiple paths.

• The source routes the single copy of each packet on different paths per packet, where the paths are selected in a probabilistic or deterministic fashion in order to aid load-balancing, and thus prolong network-lifetime. In this category, centralized [92] (the paths are calculated by the base station) and decentralized approaches [138] (calculation is done by each node independently from each other) can be further distinguished.

• The source splits the original data packet into fragments, adds some redundancy to each fragment, and then sends each fragment on one of the $n$ available paths. As it was studied in [40], if some forward error correcting code is applied that corrects $k$ ($k < n$) errors, then the method is a kind of trade-off between amount of traffic and reliability: even if some of the fragments were lost, the original message can still be reconstructed due to the added redundancy to each fragment (i.e., only $k$ fragments are needed at the destination to reconstruct the original message).

Finally, we note that if nodes use omnidirectional antennas (i.e., a single wireless transmission by a node can be received by every node within its transmission range) multi-path routing can reduce energy consumption (i.e., the availability of the routing service) in one-to-one communication over unreliable links [38].

**Route reconfiguration:** Some routing protocols forward data along a pre-established single path to save energy, and a high delivery ratio is achieved by path repair whenever a break is detected. There are two main approaches. One is that if a path break (failure) is detected, a notification is sent to the source node, which is responsible for finding an alternative path and resending the data packet (like in AODV [120]). This source-initiated approach can be expensive, if a failure occurs many hops away from the source node. Alternatively, nodes can perform path repairing locally. Here, the node having the broken link is responsible for searching alternative paths, and data is forwarded along one of these path. Although the selected alternative path may not be optimal from the view of the source node, the energy is conserved by preventing potential network floods and avoiding long-distance failure notification.

Although some routing protocols incorporate route reconfiguration, there have been proposed some localized methods (e.g., [36] and [54]), which act as separate modules, and can be used in conjunction with some routing protocols.

### 3.4.4 Security modules

These modules primarily intends to detect, and prevent or mitigate malicious faults that are caused by the adversary. Although attacks against routing can be very subtle, all of them are built upon the malicious modification or dropping of existing packets, reordering of packet sequences, and the injection of extra packets.

**Blacklisting:** Blacklisting is used to eliminate either unreliable and lossy links from the set of links used for data forwarding [55, 113, 137], or misbehaving nodes which do not follow the routing protocol (e.g., they maliciously drop, modify packets, or inject extra ones) [165].

When links are blacklisted, all nodes collect statistics about delivery rates with their neighbors, and only the links with reliability higher than a blacklisting threshold are made available for sending and receiving messages. For instance, it can be implemented in a way that each packet carries a blacklist, a minimal set of degraded-quality links encountered along its path, and the next hop is determined based on both its destination and blacklist. Alternatively, following a
decentralized approach, each node locally identifies links to be blacklisted (e.g., based on some link reliability metric described above) and drops incoming and outgoing packets on each link that it determines to have reliability below the specified blacklisting threshold. Blacklisting of misbehaving nodes is usually based on overhearing. In particular, each node continuously monitors its neighbors and checks whether they faithfully forward messages.

**Authentication:** To protect against malicious manipulations of routing messages, one can employ different cryptographic primitives. Routing protocols can guarantee source and hop-by-hop authentication for routing messages. In the former case, the origin of the message is verified at each intermediate hop and/or at the destination, while in the latter case each hop can verify the authenticity of the immediate sender (i.e., the previous hop). We further distinguish the authentication of broadcast (and multicast) and unicast data.

*Broadcast authentication:* As many routing protocols rely on flooding or broadcasting routing information, authentication of broadcast data sent by the base station (or rarely by sensor nodes) is a fundamental issue. There exist multiple techniques to achieve broadcast authentication. These include digital signature-based approaches [162] which are usually based on the optimized implementation of traditional signature schemes (like ECDSA [98, 129]), multiple message authentication based approaches [177, 189] where the origin(s) attach multiple MACs to a message from which some are verifiable by a receiver, TESLA-based approaches [100, 121] which use symmetric-key based cryptography exclusively but assume loosely synchronized clocks, and perturbation-based approaches [185] which employ perturbation polynomial based techniques.

*Unicast authentication:* The authentication of unicast data is ensured by applying conventional message authentication codes (MACs) optimized for resource-constrained sensor motes [121]. Their implementations are usually provided in the data-link layer [71, 104]. A more complex scheme using location-aware keys and MACs is proposed in [130] to provide end-to-end data authentication.

**Encryption:** Routing protocols can employ encryption to ensure confidentiality. In the topology discovery phase, it is used to conceal topology information like in [37]. In the data forwarding phase, it ensures that the message content can only be recovered by the intended receivers [130]. Similarly to unicast authentication, the implementation of required cryptographic primitives are usually already provided in the link layer [71, 104]. In the data forwarding phase, it simply prevents intermediate nodes to eavesdrop data packets [6]. A multicast encryption scheme, which supports various multicast group semantics, is proposed in [128].
<table>
<thead>
<tr>
<th>Module</th>
<th>Real-time delivery</th>
<th>Dependable delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td>Cross-layer module</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cooperative forwarding</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster-based forwarding</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Energy-based cost</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Distance-based cost</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Content-based cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link-reliability based cost</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Time-based cost</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Maintenance-based cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Probabilistic selection</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hierarchical selection</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Late selection</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Centralized selection</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Route selection towards multiple BS</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multipath selection</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Route reconfiguration</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Blacklisting</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Routing modules and their objectives.
<table>
<thead>
<tr>
<th>Module</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-layer module</td>
<td>MACRO [50], SIGF [165], CCMR [133]</td>
</tr>
<tr>
<td>Cooperative forwarding</td>
<td>CBF [20], RBP [142], DRB [74], AsOR [163]</td>
</tr>
<tr>
<td>Cluster-based forwarding</td>
<td>MACRO [50], SIGF [165], CCMR [133], Energy Aware Routing [138], GBR [136], TEEN [106], APTEEN [107], PEGASIS [97], GEAR [182], MECN [132], TTDD [176], SAR (DAM) [45], HPAR [93], RPAR [26]</td>
</tr>
<tr>
<td>Energy-based costs</td>
<td>MACRO [50], SIGF [165], DAMER [38], CCMR [133], Energy Aware Routing [138], GBR [136], TEEN [106], APTEEN [107], PEGASIS [97], GEAR [182], MECN [132], TTDD [176], SAR (DAM) [45], HPAR [93], RPAR [26]</td>
</tr>
<tr>
<td>Distance-based costs</td>
<td>GOAFR [81], GPSR [72], GEDIR [144], GPSVR [90], GDSTR [89], MACRO [50], SIGF [165], CCMR [133], GEAR [182], BVR [51], GLIDER [46], MAP [14], VPCR [115], MECN [132], SPEED [44], MMPSPEED [48], VCap [21], ABVCap [154], GFG [12], Hop ID [187], NADV [88], LCLR [77], CLDP [76], ProgressFace [95], VirtualFace [153], RPAR [26], EFS [137]</td>
</tr>
<tr>
<td>Content-based costs</td>
<td>Directed Diffusion [66], GBR [136], IDSQ/CADR [27], Secure DD [174]</td>
</tr>
<tr>
<td>Link-reliability based costs</td>
<td>MACRO [50], SIGF [165], DAMER [38], CCMR [133], NADV [88], EFS [137]</td>
</tr>
<tr>
<td>Time-based costs</td>
<td>TinyLUNAR [117], Secure-TinyLUNAR [31], SPEED [44], MMPSPEED [48], RPAR [26]</td>
</tr>
<tr>
<td>Maintenance-based costs</td>
<td>MER [8]</td>
</tr>
<tr>
<td>Path selection</td>
<td>ARRIVE [70], SIGF [165], Rumor Routing [13], Energy Aware Routing [138], ACQUIRE [43], MMPSPEED [48]</td>
</tr>
<tr>
<td>Probabilistic selection</td>
<td>TEEN [106], APTEEN [107], PEGASIS [97], MECN [132], TTDD [176], SAR (DAM) [45], HPAR [93]</td>
</tr>
<tr>
<td>Hierarchical selection</td>
<td>MCFR [175]</td>
</tr>
<tr>
<td>Late selection</td>
<td>HPAR [93], INSENS [37]</td>
</tr>
<tr>
<td>Centralized selection</td>
<td>INSENS [37], Colored Tree [151], TTDD [176]</td>
</tr>
<tr>
<td>Route selection towards multiple BS</td>
<td>INSENS [37], Colored Tree [151], TTDD [176]</td>
</tr>
<tr>
<td>Multipath selection</td>
<td>ARRIVE [70], INSENS [37], Colored Tree [151], SIGF [165], Secure DD [174], Energy Aware Routing [138], Directed Diffusion [66], GBR [136], MMPSPEED [48]</td>
</tr>
<tr>
<td>Route reconfiguration</td>
<td>MT [164], Secure DD [174], Directed Diffusion [66], GBR [136], MECN [132], TTDD [176], SPEED [44]</td>
</tr>
<tr>
<td>Blacklisting</td>
<td>ARRIVE [70], SIGF [165], EFS [137]</td>
</tr>
<tr>
<td>Authentication</td>
<td>INSENS [37], SIGF [165], Secure DD [174], Secure-TinyLUNAR [31]</td>
</tr>
<tr>
<td>Encryption</td>
<td>INSENS [37], SIGF [165]</td>
</tr>
</tbody>
</table>

Table 2: Modules and their implementations.
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Base station</th>
<th>Network model</th>
<th>Sensor nodes</th>
<th>Operational model</th>
<th>Communication pattern</th>
<th>Reporting model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num.</td>
<td>Mobility</td>
<td>Presence</td>
<td>Deployment</td>
<td>Query</td>
<td>Response</td>
</tr>
<tr>
<td>MCFA [175]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Energy Aware Routing</td>
<td>More</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>Directed Diffusion [66]</td>
<td>More</td>
<td>Limited</td>
<td>Continuous</td>
<td>*</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>GBR [136]</td>
<td>More</td>
<td>Limited</td>
<td>Continuous</td>
<td>*</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>TEEN [106]</td>
<td>One</td>
<td>Fixed</td>
<td>Continuous</td>
<td>Random</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>APTEEN [107]</td>
<td>One</td>
<td>Fixed</td>
<td>Continuous</td>
<td>Random</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>PEGASIS [97]</td>
<td>One</td>
<td>Fixed</td>
<td>Continuous</td>
<td>Random</td>
<td>Data ID, Location</td>
<td>×</td>
</tr>
<tr>
<td>MECN [132]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>Location</td>
<td>×</td>
</tr>
<tr>
<td>SAR (DAM) [45]</td>
<td>More</td>
<td>Limited</td>
<td>*</td>
<td>*</td>
<td>×</td>
<td>ID</td>
</tr>
<tr>
<td>HPAR [93]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>×</td>
</tr>
<tr>
<td>TinyOS Beaconing [62]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>×</td>
<td>ID</td>
</tr>
<tr>
<td>TinyLUNAR [117]</td>
<td>More</td>
<td>Mobile</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ID</td>
</tr>
<tr>
<td>Secure-TinyLUNAR [31]</td>
<td>More</td>
<td>Mobile</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ID</td>
</tr>
<tr>
<td>INSENS [37]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ID</td>
</tr>
<tr>
<td>Secure DD [174]</td>
<td>More</td>
<td>Limited</td>
<td>Continuous</td>
<td>*</td>
<td>Data ID</td>
<td>×</td>
</tr>
<tr>
<td>ARRIVE [70]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>×</td>
<td>ID</td>
</tr>
<tr>
<td>MT [164]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ID</td>
</tr>
<tr>
<td>DAMER [38]</td>
<td>One</td>
<td>Fixed</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>ID</td>
</tr>
</tbody>
</table>

Table 3: The operational and network model of each module implementation.
3.5 Summary and future work

Existing surveys on sensor network routing hardly support the development of sensor network applications due to their rough operational and network models. Moreover, they tend to neglect dependability concerns as well as routing modules which function only as a part of a routing protocol. In this section, we proposed a modular approach to design routing protocols for sensor network applications, where a routing protocol is a combination of different routing modules and each module has some routing objectives. Following this approach, the main steps of designing a routing protocol are as follows: (1) identification of the routing objectives, (2) selection of routing modules based on the identified objectives such that a module from each core component must be selected (Table 1), (3) identification of the network and operational model of the application, (4) selection of specific module implementations based on the identified network and operational model (Tables 2 and 3), (5) integration of selected implementations.

We anticipate that the routing protocol of most sensor network applications (including the target applications of the WSAN4CIP project) can be designed by using our modular approach. Although the available module implementations are extensive, it may occur that a new implementation of a specific module will be required within the project. This is because the foreseen applications are expected to have unusual network model (like nodes with unconstrained power supply, or they form a special network topology) where more efficient implementations can be used than the available ones. Nevertheless, it is unlikely (and we intend to avoid) that all identified modules will need a new (yet unavailable) implementation, or a novel protocol will needed to be designed using a clean-slate approach.
4 Dependability of clustering protocols

To support scalability and increase the lifetime of WSNs, the sensor nodes are often grouped into clusters. These clusters are mostly disjoint and non-overlapping, and consist of a leader, referred to as the cluster head (CH), and member nodes. The CHs are elected in the network by usually using some local or global election mechanism, for instance, nodes with the highest node degree in their neighborhood or closest location to given reference points in the network can become cluster head. On the other hand, member association is done via some association procedure, e.g., selecting CH with the shortest communication distance and belonging to its cluster. The cluster head controls the working of its cluster in the way that periodically collects sensor readings from the associated member nodes and forwards the gathered information to a (central) sink station often after applying aggregation or filtering (or other local processing).

In this section, we give a critical overview of the state-of-the-art clustering protocols and their cluster head election mechanisms in wireless sensor networks focusing on dependability issues. First, we define the taxonomy of dependable cluster head election and clustering. We identify the properties of dependability, such as termination, functionality, role non-manipulability, association non-manipulability, role unpredictability, association unpredictability, role un-identifiability and association un-identifiability, and give their definitions. We also provide a relationship graph reflecting the natural dependencies of these properties. Moreover, we define three adversary models, such as passive, active and compromising, that cover the possible failure and attack scenarios. Then, we shortly describe the most relevant clustering protocols and their cluster head election mechanisms, and systematically analyze them investigating the fulfilment of the defined dependability properties in case of the different adversary models.

4.1 Dependability properties

In order to make the existing cluster head election solutions comparable, we define a set of properties that are, in our opinion, important to satisfy in case of a dependable cluster head election protocol. The set of properties contains basic properties (e.g., termination) and also advanced properties (e.g., non-manipulability). The definition of the properties rely on the following notions:

**Role** - The role of a sensor node is either cluster head or cluster member.

**Cluster head** - A sensor node is called cluster head, if it has been elected as cluster head during the election process.

**Cluster member** - A sensor node is called cluster member, if it has not been elected as cluster head, but it is associated with a cluster head.

**Association** - The association of a cluster member node is the knowledge of which cluster (and cluster head) it belongs to.

Based on these notions we define the properties that a dependable clustering solution has to fulfill as follows.

**Termination:** after a finite amount of time the protocol terminates (i.e., every node knows that the protocol ended)

**Functionality:** a protocol is functional if it satisfies the properties of termination, completeness, and consistency, where completeness and consistency are defined as follows:

- **Completeness:** every node for which the protocol terminates knows its role
- **Consistency:** the cluster head of every cluster member is in the set of cluster heads

**Role non-manipulability:** for every node for which the protocol terminates the attacker cannot influence the node’s role

**Association non-manipulability:** for every cluster member the attacker cannot influence the node’s association
**Role unpredictability:** from the viewpoint of the attacker every node has the same probability to become cluster head

**Association unpredictability:** if the roles are predictable then from the viewpoint of the attacker all the cluster head nodes have the same probability to have the given member node associated with it

**Role un-identifiability:** for every node the attacker cannot identify the node’s role during the cluster head election process

**Association un-identifiability:** for every node for which the attacker knows that it is a cluster member the attacker cannot identify which cluster head it is associated with during the cluster head election process

The termination and functionality properties are basic requirements that all the cluster head election approaches have to fulfill, not just the solutions that aim at dependability. The termination property requires the protocol to terminate in a finite amount of time, i.e., dead-locks or infinite waiting times are not allowed. The functionality property can be interpreted only if the termination property is met, and it requires every terminated node to have its role. If this role is cluster member, then the node should have a cluster head that also believes it is a cluster head. This means that if a cluster member node thinks that it is associated with node A, and node A does not believe it is a cluster head, then the protocol is not functional.

The following properties, i.e., non-manipulability, unpredictability and un-identifiability, are special features of dependable cluster head election mechanisms. These properties all require the functionality property to be met, since without functionality it makes no sense to speak about more advanced requirements. The two non-manipulability properties state that the adversary should not be able to alter the role or the association of the sensor nodes during the election process. This is of paramount importance, because if an adversary could do that, then he could, for example, force a node controlled by himself to become cluster head all the time. Thus, he could take over the control of a significant part of the sensor network, i.e., the whole cluster of that cluster head. Alternatively, the adversary could also force its node to become always cluster member, thus saving energy.

The unpredictability and un-identifiability properties are somewhat similar, there is, however, a temporal difference between them. Namely, unpredictability requires the adversary to be unable to precisely estimate the identity of the cluster head in advance, while un-identifiability requires the same with the relaxation that the adversary can observe the election process. Both of these requirements aim at preventing the adversary from identifying the upcoming cluster head, since it may become the primary target. Note that in this survey, we investigate the identifiability of the upcoming cluster head node only before and during the election, since we focus on the dependability of the election process itself.

As we can see, the above properties inherit some natural dependencies that are illustrated in Figure 4 in a so called "relationship graph". In the graph, \( A \rightarrow B \) means that the fulfillment of \( B \) requires the fulfillment of \( A \).

![Figure 4: Relationship graph of dependability properties of clustering protocols](image-url)
4.2 Adversary models

We categorize the strength of the adversary into three typical categories as follows.

**Passive adversary:** A passive adversary is only able to perform eavesdropping at any point of the network, or can even eavesdrop the whole communication of the network. As sensor nodes use wireless channels for their communication, this is not a far-fetched assumption. This adversary is also able to do traffic analysis or any kind of statistics based on the eavesdropped information.

**Active adversary:** An active adversary possesses the abilities of a passive adversary, but he also can tamper with the wireless channel. This means, that an active adversary can replay, forge, modify, and delete messages beside eavesdropping. However, the active adversary is still an outsider adversary, i.e., he does not know the secret keys (if any) of the sensor nodes.

**Compromising adversary:** The most powerful adversary we consider is the compromising adversary who possesses all the abilities of an active adversary, moreover, he also knows the secret information stored in the sensor node (e.g., keys) since he is able to compromise the node. The compromising adversary is thus an insider adversary that can also change the inner state of the node. Compromised nodes can behave like regular nodes, but are able to change their behavior according to the adversary’s willing at any time. Therefore, the compromised adversary represents the Byzantine attacker model and also the Byzantine fault model.

Note that in case a compromising adversary, the compromised node cannot be required anymore to fulfill any of the above defined properties. Therefore, in case of node compromise, we investigate the influence of the compromised node on the remaining (i.e., not compromised) part of the network, and evaluate the properties only on that part.

As it already follows from the description of the adversarial models, there are also inherent dependencies that we illustrate in Figure 5. Here again, $A \rightarrow B$ means that the fulfillment of a property assuming $B$ requires the fulfillment of the same property assuming $A$.

4.3 Analysis of state-of-the-art clustering protocols

In this subsection, we analyze the state-of-the-art clustering protocols and their cluster head election mechanisms in WSNs investigating the fulfillment of the defined dependability properties in case of the different adversary models.

In this analysis, we considered basically more than 20 papers. During the first round we eliminated almost half of the papers from further investigations, namely \[3, 9, 23, 25, 33, 49, 99, 102, 110, 139, 146, 149, 152, 160\], because either they do not propose any cluster head election mechanism only discuss the application of clustering for given purposes, or they describe just the basic idea of a CH election method but do not specify the detailed protocol.

In case of the remaining papers, we systematically investigated the fulfillment of the defined dependability properties assuming the passive, active and compromising adversary model, respectively. To simplify the investigation we followed the property relationship graph (see above). We...
found that the protocols in these papers basically show weak dependability. They can be grouped in two categories according to the property pattern they show. Hence, the protocols proposed in [10, 16, 35, 58, 158, 180, 181, 183] satisfy the termination, functionality, role and association non-manipulability properties assuming passive adversary, and only the termination property\(^2\) assuming active and compromising adversary (property pattern 4-1-1, see Figure 6.a). The protocols introduced in [61, 80, 112, 140] satisfy also the role and association unpredictability properties in case of passive adversary (property pattern 6-1-1, see Figure 6.b). In the lack of authentication and security mechanisms none of these protocols can guarantee dependable working in case of active and compromising adversary.

In the following, we describe and analyze 5-5 protocols in both categories in detail. The first five protocols below belong to the class of protocols that show the property pattern 4-1-1, while the remaining five protocols belong to the class 6-1-1.

### 4.3.1 PANEL

**Description:** PANEL is a position-based cluster head election protocol for wireless sensor networks proposed in [16]. One of the main assumptions that PANEL relies on is that the sensor nodes are static and they are aware of their geographical position. The authors note, however, that PANEL does not need precise position information, therefore, the inaccuracy of the positioning mechanism does not limit its applications. It is further assumed that the sensor nodes are deployed in a bounded area which is partitioned into fix geographical clusters. The clustering is determined before the deployment of the network, and each sensor node is pre-loaded with the geographical information of the cluster which it belongs to. The density of the network is assumed to be large enough such that the nodes within each cluster are connected when they use maximum power for transmission. In other words, there exists a route between any pair of nodes of a given cluster that contains only sensors from that cluster. Finally, time is assumed to be divided into epochs, and the nodes are assumed to be synchronized such that each of them knows when a new epoch begins.

The protocol aims at electing a single cluster head per cluster. For this, at the beginning of each epoch, a reference point is computed in each cluster by every node in the cluster in a completely distributed manner. In fact, the computation of the reference point depends only on the epoch number, and it can be executed by every node independently and locally. Once the reference point is computed, the nodes in the cluster elect the node that is the closest to the reference point as the cluster head for the given epoch. For the election each node sets a timer, the expiration time of which

\(^{2}\text{Note that even if the fulfillment of the termination property is not clear from the protocol description we still assume it in case of all the 3 adversary models.}\)
is proportional to the distance between the node’s position and the reference point of its cluster. When this timer expires, the node broadcasts a message with maximum power in which it announces itself as the cluster head unless the node heard such an announcement from another node before its timer expired. The announcement contains the current epoch number, the identifier and the position of the originator of the announcement. When a node hears an announcement, it verifies if the originator of the announcement is closer to the reference point than the node known to be the closest so far (which can be the node itself if it has not heard any announcement yet). If so, then the node records the originator of the announcement as the candidate cluster head, and re-broadcasts the announcement. Moreover, if the node still has its timer active, then it cancels it. Otherwise, the node silently discards the announcement. Announcements that belong to other clusters are also discarded in order to limit the propagation of an announcement within the cluster that it is concerned with.

In [17], the authors of PANEL propose some security extensions to the original protocol that aim at increasing the resistance of the protocol against spoofing attacks and manipulation of the outcome of the cluster head election. First of all, announcement messages can be authenticated with a broadcast authentication scheme in order to detect the use of invented identifiers of an outsider adversary. Second, in PANEL, the nodes keep in their routing tables the position information of the other nodes from which they have already heard an announcement. Therefore, on the one hand, the nodes can detect if a compromised node, i.e., insider attacker, tries to report itself at different positions in different epochs, and, on the other hand, it is also detectable if a compromised node generates multiple identities for itself.

Analysis: PANEL satisfies the termination property, because the cluster head election phase always ends after a predefined time \( T \), moreover, at the end of the election phase each node either considers itself cluster head, because it sent a cluster head announcement, or considers itself a cluster member, because it received such an announcement from a node that is closer to the reference point. As the nodes decide to stop based on their local timers, even an active attacker cannot prevent them from terminating the cluster head election phase.

However, PANEL satisfies the functionality property only in case of a passive attacker. On the one hand, passive attackers cannot interfere with the execution of the protocol, and in the absence of such interference, each node computes the same reference point (given the assumption on clock synchronization) and receives the cluster head announcement message of the node that is the closest to this reference point (given the assumption on position awareness and connectivity within the cluster). Hence, the same node will be considered cluster head by all nodes in the cluster. On the other hand, an active attacker can bring the system into an inconsistent state. Let us assume, for instance, that two nodes, say \( i \) and \( j \), both sent cluster head announcements nearly at the same time. Let us further assume that \( j \) is closer to the reference point than node \( i \), but a third node, say \( k \), first received the announcement of \( i \). If the attacker jams node \( k \) and prevents the reception of the announcement of node \( j \) then node \( k \) will consider \( i \) as cluster head, while after receiving \( j \)’s announcement \( i \) will not consider itself cluster head.

Similarly, PANEL ensures non-manipulability assuming a passive attacker, because such an attacker cannot influence the reference point computation and the execution of the cluster head election protocol. At the same time, an active attacker can always coerce a node to elect itself cluster head by preventing it to receive any messages, hence, it can manipulate the outcome of the protocol.

Finally, even a passive attacker can predict which nodes will be cluster heads, as it can compute the reference point in any epoch and can know or learn (from eavesdropped cluster head announcements) the positions of the sensor nodes in the cluster.

4.3.2 Fault-tolerant clustering of wireless sensor networks

Description: In [58], the authors propose a mechanism to detect cluster head failures and recover sensors from a failed cluster without re-clustering the network. The authors assume that some high-energy nodes called gateways are deployed in the network. Cluster heads can be only gateways, which are directly connected to each other. The regular nodes (sensors) and gateways are stationary and know their geographic location by using a GPS system. To provide MAC layer communication a
TDMA protocol is applied which uses time cycles. A cycle consists of three types of slots. First, the gateways inform the sensors about the TDMA schedule and the routing information in a Route Update slot. Second, the gateways allocate Sensor Data slots for the sensors to send data. Third, the gateways exchange their status in a Status Update slot, which also acts as a heartbeat message.

The gateways create different sets to organize the network into clusters. Each gateway creates a range set called RSet. A sensor s belongs to an RSet of gateway g if and only if s and g are in each other’s transmission range. Since every node knows its exact location, the gateway nodes can select their cluster members from the RSet having the minimum communication cost. For recovery purposes each gateway constructs another set called backup set (BSet) containing nodes that do not belong to the cluster of the gateway but are included in its RSet. If a sensor has to be recovered every gateway checks its own BSet for the sensor. If the sensor is in multiple BSets the gateway with the minimum communication cost will be chosen as the new cluster head and informs the sensor about the new association.

**Analysis:** During the clustering mechanism the gateways create a range set and associate sensors with themselves by minimizing the communication cost, in addition, they create a backup set for the case of recovery. The designers of the protocol do not tell too many details about how the protocol runs, hence it is not clear that the protocol terminates in a given time \( T \) if an active attacker wants to prevent it. However, we assume that the protocol satisfies the termination property for the further examinations (it can be easily achieved by using a timeout mechanism).

The protocol satisfies also the functionality property in case of a passive attacker (completeness and consistency are given, because roles are pre-determined). However, the proposed clustering mechanism is only giving the association between the sensors and the gateways, thus inconsistency can be achieved by an active attacker. When a sensor s sends its location information to the neighboring gateways an attacker can overwrite the location information in the message sent to the closest gateway g. Hence, every gateway but g believes that g is the closest gateway to s. Moreover, the attacker can associate s with a gateway h that locates beyond the sensor’s transmission range. So h will act as a cluster head of s but it will never receive any messages from s.

The protocol satisfies the non-manipulability property in case of a passive attacker, because a passive attacker cannot influence the association process. On the other hand, an active attacker can easily influence the associations by modifying the location information in the messages.

Based on the location information, even a passive attacker can predict the association between sensors and gateways. Hence, the unpredictability properties cannot be assured.

### 4.3.3 A dependable clustering protocol for survivable underwater sensor networks

**Description:** In [159], a dependable clustering protocol is proposed to provide a survivable cluster hierarchy against cluster head failures in underwater sensor networks (UWSNs). The proposed algorithm is applicable to any wireless sensor networks not just to UWSNs. The protocol selects a primary and a backup cluster head for each cluster member to avoid re-clustering in case of failure. The network consists of homogenous sensor nodes, which are in one of the following three states during the clustering mechanism: cluster head, cluster member and cluster head candidate state. Initially, each node is in the cluster head candidate state.

The protocol is divided into three phases, such as initialization, clustering and finalization phase. In the initialization phase, each node discovers its one-hop neighbors, and maintains an uncovered neighbor set which contains these neighbors being still in cluster head candidate state. From this set, each node generates all the possible combinations of clusters, which means that if a node has \( n \) uncovered neighbors it will generate \( 2^n \) potential clusters. Among all potential clusters, a candidate selects a cluster as its qualified cluster with minimum average cost, which is computed from a given function based on energy consumption and the residual energy.

In the clustering phase, each candidate sends the average cost of its qualified cluster to all candidates within its two-hop range, and collects the average costs sent by the other candidates. Candidates with the minimum average cost within their two-hop range will change their status to cluster head and send an invite message to the members of their qualified cluster. Candidates, which received
the invite message, change their status to cluster member and as an acknowledgement to the invite message, broadcast a join message and add the sender of the invite message to their cluster head list as their primary cluster head. If no invite message is received in a given time period, the candidate will stay in the same state and reselects its qualified cluster, because some nodes in it might have changed their status. After a node becomes a cluster member, it will keep monitoring the invite messages for other nodes, and adds the sender of them to its cluster head list. Clustering will be repeated until all candidates become either a cluster head or a cluster member.

To guarantee that every cluster member has at least two elements in its cluster head list, a finalization phase is run. In this phase, all cluster members return to cluster head candidate state, and the clustering protocol is performed one more time. This results that each cluster member will be covered by at least one more cluster head or will become a cluster head itself. Combining the two cluster head lists excluding the primary cluster head, we get the backup cluster head list, in which the cluster head with the minimum distance to the cluster member will be selected as the backup cluster head.

**Analysis:** After a given time $T$, the protocol will terminate, even assuming an active attacker. However, an active attacker can replay the average cost message with a very small cost value and delay the termination. If the attacked node did not receive any invite messages, it stays in cluster head candidate state, and reselects its qualified cluster. If a predefined timeout $t_1$ is reached, the whole clustering mechanism stops and nodes which are still in cluster head candidate state will represent themselves in the network.

The functionality property is satisfied only in the case of a passive attacker. An active attacker can bring the network into an inconsistent state by fabricating and replaying invite and join messages. If a candidate $i$ declares itself as a cluster head and sends an invite message to the members of its qualified cluster, the attacker can jam the invited nodes. After it, the attacker replays a fabricated join message to $i$ on behalf of an invited node $j$. Now, $i$ guesses that $j$ is its cluster member, but $j$ is still in cluster head candidate state in fact. Since other nodes guess $j$ as a candidate, they can invite it to be their cluster member.

The protocol is non-manipulable by a passive attacker, however, an active attacker can easily manipulate roles and associations, as we described previously.

Based on the given cost function, even a passive attacker can predict the roles and the associations of the nodes. Hence, the unpredictability properties cannot be assured.

### 4.3.4 Clique Covering

**Description:** Clique Covering or CC [10] is a clustering protocol for wireless sensor networks which includes a fail-safe procedure for dealing with node failure or removal. The protocol provides mechanisms for cluster head election, clustering the nodes into cliques, backbone formation and backbone maintenance in case of node failure or removal.

In the cluster head election and cluster formation phase, the nodes elect the most suitable ones among themselves as cluster heads and form cliques headed by the elected nodes. The election is based on the weight of the node and its unique identifier (ID). The weight is a real number $\geq 0$ which depends on the node’s current status and application requirements indicating the suitability of the node for acting as cluster head. The higher the weight the better is the node for taking the cluster head role. The protocol is executed at each node in such a way that the node decides its role (cluster head or cluster member) as soon as its neighbors with higher weight values have decided their own role (it is assumed that every node knows its own unique ID, its own weight and the ID and weight of each of its neighbors).

CC is started by the *init nodes* having the biggest weight among all their neighbors. The init nodes send a broadcast message advertising themselves as cluster heads. Upon receiving this message, the adjacent nodes exchange information about their own neighbors with the sender. Based on this information, the cluster heads select all those neighbors that can be associated to their own cluster while maintaining the clique property, and invite them to join their cluster. Every node that has not

---

3Cliques are clusters in which the nodes are directly connected to each others.
been invited into any cluster decides to be a cluster head itself. The protocol terminates when every node belonging to a cluster is either a cluster head or a cluster member and knows the role and the cluster head of all its neighbors.

After the cliques are created the formation of a connected backbone starts. Every node knows the ID and weight of each neighbor as well as the ID and the weight of the cluster head to which each neighbor is affiliated. Based on this information each node computes paths to cluster heads one and two hops away and sends the results to its own cluster head. For each cluster head one path is selected. Once a cluster head has determined the connection information in its cluster, it can identify the best paths to all the cluster heads that are at most three hops away (this is the necessary and sufficient condition to form a connected backbone). Finally, cluster heads three hops away are joined by selecting two intermediate nodes between them. When a cluster head has calculated the routes to all the other cluster heads up to three hops away, it disseminates this information in its cluster. Therefore, each node knows if it is used as gateway to interconnect two adjacent clusters, and how to reach neighboring cluster heads.

CC also provides a backbone reorganization procedure when a backbone node fails or disappears. The problem is handled locally by the adjacent nodes. If the failed or removed node is a cluster head a new cluster head node will be elected in the cluster and the backbone will be 'repaired' by selecting new gateway node(s) if it is necessary. Otherwise, only the failed gateway node will be replaced by new gateway node(s).

Analysis: We assume that CC satisfies the termination property. As mentioned, the protocol terminates when every node belonging to a cluster is either a cluster head or a cluster member and knows the role and the cluster head of all its neighbors. Although, the authors haven’t discussed what procedures, e.g. using timeouts, are implemented to ensure termination, we still assume that CC incorporates some mechanism to handle this issue.

With regard to the functionality property, CC satisfies it but only in case of a passive attacker. Such an adversary cannot interfere with the execution of the protocol and thus cannot influence the election process. The decision about the node’s role is guaranteed if termination of the protocol is ensured. However, an active adversary can bring the system into an inconsistent state by, for example, propagating invalid weight value (it is not clear how the weight values are known for the nodes but we assume some propagation mechanism in place).

CC guarantees non-manipulability assuming a passive attacker because such an attacker cannot influence the execution of cluster head election. However, an active adversary can always send fake messages or manipulate others’ communication influencing cluster formation.

Unpredictability is not given in CC because the weights can be known to the adversary by, for instance, eavesdropping the messages when neighbors and weights are discovered. Since the execution of the protocol is basically deterministic, being aware of the node weights and adjacency relations the roles and associations can be predicted.

4.3.5 SANE based on Merkle’s puzzle and homomorphic encryption

Description: The authors of SANE [140] propose three solutions for non-manipulable cluster head election. Two of them can also ensure the unpredictability of the election result. The first proposed solution applies Merkle’s puzzle [111] for setting up pairwise keys between the current cluster head and the cluster members, and an additively homomorphic encryption transformation [22] for encrypting and summing up the pseudorandom contributions of the sensor nodes. These contributions together define the ID of the next cluster head.

The protocol works as follows. At the end of each epoch, the current cluster head $A_t$ applies Merkle’s puzzle to establish pairwise keys $k_i$ with each sensor nodes $s_i$ (including itself). Then, each node computes the homomorphically encrypted sum $\sum_{j=1}^{i} E_{k_j}(r_j) = E_{k_0}(r_1) + \sum_{j=1}^{i-1} E_{k_j}(r_j)$ starting at $s_1$ with the sum equal to $E_{k_1}(r_1)$. After contributing to the sum, node $s_i$ adds itself to the list of contributors, and unicasts the sum to node $s_{i+1}$ reliably. If $s_{i+1}$ has failed, $s_i$ unicasts the sum to the next available node. The process ends when all nodes including the current cluster head have contributed to the sum, and the sum is available to all sensor nodes. After a predefined short time in
the decryption phase, the current cluster head floods all the pairwise keys $k_j$ and the corresponding ID’s to all nodes. Each node then individually decrypts the ciphered sum using the pairwise keys that results in $R_i = \sum_j E_k(r_j) - \sum_j k_j$. In case $s_i$ does not receive key $k_j$, it explicitly requests it from $A_k$. If $A_k$ persistently fails to deliver $k_j$, $s_i$ proceeds with processing the encrypted sum without considering $k_j$. Finally, a mapping function is applied at each node in order to convert the random aggregate value $R_i$ to the ID of the upcoming cluster head.

**Analysis:** This proposed SANE solution satisfies the termination property, since each node considers the cluster head election to be finished right after the decryption operation and the subsequent mapping. These latter operations are performed independently at each node, and terminated regardless of message losses or passive attacks. Moreover, even an active attacker cannot interfere with the termination of the algorithm, as nodes decide to finish based on their local operation. For the latter to hold, however, one has to assume that each node has a local timer, at the expiration of which the node terminates the algorithm independently of its current state.

The functionality property is satisfied in case of a passive attacker, however, an active attacker can mislead a node regarding the cluster head it elects. A passive attacker cannot circumvent the establishment of the pairwise keys, neither can prevent the encrypted sum to be unicast along the row of sensor nodes. On the contrary, an active attacker can interfere with the election process and imply different cluster heads for a set of nodes. For example, an active attacker can jam node $k$ in the decryption phase, thus, $k$ will not receive some of the pairwise keys, which will result in an altered outcome of the decryption and the mapping at node $k$. This means that $k$ will elect a cluster head different from the cluster head elected by the remaining nodes. Although the authors of [140] accept this situation as normal operation, it does not satisfy our functionality property, as the cluster head elected by the jammed node does not consider itself cluster head (i.e., it is not in the set of cluster heads).

The non-manipulability properties of the algorithm are assured assuming a passive attacker, because such an attacker cannot influence the computation of the ciphered sum and the subsequent operations. At the same time, an active attacker can always coerce a node to elect itself cluster head by preventing it to receive any messages, hence, it can manipulate the outcome of the protocol as described previously.

The fulfillment of the unpredictability properties is an important feature of the solution at hand. A passive attacker cannot a priori determine the ID of the upcoming cluster head (role unpredictability), since the sensors contribute to the encrypted sum by a random value (i.e., the process is not deterministic). Moreover, the sum aggregating the contributions is always encrypted, thus, an attacker cannot learn anything about the contributions until the decryption phase when the keys are revealed. Since the role of the nodes is unpredictable, the algorithm also satisfies the association unpredictability property.

**4.3.6 SANE based on a commitment scheme**

**Description:** The other unpredictable solution proposed by the authors of [140] relies on a commitment scheme, where the nodes participating in the election process first have to commit to a random value, and a short time later all the nodes reveal their random value. In this way, all the nodes contribute to a common value calculated as the sum of the nodes’ random values. This common value is used then to identify the upcoming cluster head node. The commitment is necessary for ensuring that a malicious node does not change its contribution after the random values are revealed, since the other nodes can check whether the commitment corresponds to the revealed random or not.

The operation of the commitment based SANE algorithm is as follows. In the first phase, called the commitment phase, each sensor node $s_i$ commits to a random value $r_i$ with $c(r_i)$, where $c(x)$ denotes the commitment on $x$ that does not reveal $x$ itself. $s_i$ sends its commitment to all other nodes. Later in the revealment phase, nodes reveal their random values $r_i$ by sending it to all other nodes. Each receiving node $s_i$ checks whether the commitment $c(r_j)$ from node $s_j$ verifies for $r_j$. Finally, the nodes sum up the verified random values as $R_i = \sum_j r_j$, and apply a mapping function in order to convert $R_i$ to a node ID that appoints the upcoming cluster head.
### Analysis
Similarly to the previous SANE approach, the commitment scheme based SANE approach also terminates both in case of passive and active attacks. Each node performs its calculations independently of the other nodes, and if we assume that there are timeouts for the commitment and the revealment phases, then every node will finish the algorithm at latest when the corresponding timer expires. Message losses cannot prevent the algorithm from termination, since lost messages are simply neglected.

However, lost messages or alternatively an active attack (e.g., jamming) can interfere with the fulfillment of the functionality property. Some nodes may not receive all the commitments, or later, all the valid random values transmitted by the nodes. This would result in nodes possibly electing different cluster heads. According to our definitions, this state violates the consistency property, since some nodes may elect such a node as cluster head that does not consider itself cluster head. Consequently, the functionality property does not hold in case of an active attack. In general, an active attacker can deny the election of a targeted node by selectively suppressing messages sent by honest nodes. All the same, in case of a passive attack the functionality property is satisfied, since under optimal conditions (i.e., no message losses, etc.) the algorithm is able to elect the same cluster head at each node (the authors of [140] call this the agreement property).

In case of a passive attack, the non-manipulability property of the algorithm is assured, because a passive attacker cannot interfere with the commitments, thus, it cannot influence $R_i$. However, an active attacker can always use jamming in order to prevent a node to receive any messages, hence, it can manipulate the outcome of the protocol for that particular node.

The unpredictability of the cluster head election result in case of the commitment based SANE approach is again a property to emphasize. The nodes commit to a previously unknown random value in the commitment phase, thus, the attacker is not able to predict the outcome. The commitment flooded in the network does not reveal the original random value, which means that the attacker cannot learn anything from the received commitments. This means that the role unpredictability property is satisfied by the algorithm, which implies that the association unpredictability property is also satisfied.

### 4.3.7 Fault-tolerant clustering in ad hoc and sensor networks

**Description:** In [80], the authors propose two clustering algorithms for general graphs and for unit disk graphs (UDG), from which we investigated the second one. It is assumed that the nodes are homogenous and communicate by a synchronous model, where time is divided into rounds. Every message has a restricted size of $O(\log n)$ bits and can contain only a constant number of node identifiers. The algorithm can be divided into two phases. In the first phase, a dominating set of cluster heads is selected, since in the second phase this set is extended to a $k$-fold dominating set where every node is covered by at least $k$ cluster heads.

The first part of the algorithm works by repeatedly decreasing the number of active nodes, which are the potential cluster heads. Initially, all nodes are active. In each round, an active node $v$ considers only those active neighbors that are within distance $\Theta$ of $v$. The range $\Theta$ starts with a small value and is doubled in every round. Hence, it is implicitly assumed that nodes can sense their neighbors in a given distance. In the $i^{th}$ round $r_i$, every active node chooses a random identifier (ID$_i$) in the range of $[1...n^4]$, which ensures with high probability that the IDs will be unique. Every active node elects among its active neighbors (in the given range $\Theta$) the one with the highest ID (possibly itself). If an active node is elected at least by one active node, it remains active. If not, it becomes passive and stops executing the first part of the algorithm. Nodes, which remain active after the last round will become cluster heads.

In the second part of the algorithm, the number of neighboring cluster heads of all nodes is examined. If that number $c$ is less than $k$, the given node elects randomly $k - c$ neighboring regular nodes to be its cluster head and informs these nodes about it.

**Analysis:** The first part of the algorithm consists of a given round, while the second part is only one more round. Since every node executes the algorithm independently from other nodes, the execution
depends only on the inherent state of the nodes. Therefore, it is guaranteed, that the protocol terminates in time $T$, even in case of an active attacker.

However, the functionality property is satisfied only by assuming a passive attacker. An active attacker can modify the ID sent by a node $v$ to a high value and jam the sent election messages towards $v$. Hence, $v$ does not know about the election and changes its status to passive, while its neighbors guess it is still active. If this happens in the last round, the passive neighbors of $v$ will consider $v$ as their cluster head. Moreover, if an attacker jams the messages in the second phase of the protocol, in which a node $v$ informs its $k - c$ neighbors to be its cluster heads, $v$ will consider them as cluster heads, while they do not know about it. So, the system can be brought into an inconsistent state.

Roles and associations can be also manipulated by an active attacker by modifying IDs, while a passive attacker trivially cannot manipulate them.

The cluster head election process is based on random numbers generated in all rounds in a distributed way. Therefore, a passive attacker cannot predict the status of the nodes beforehand, only when nodes disclose their randomly chosen IDs. Moreover, the attacker cannot be sure that passive nodes will be regular nodes after the first phase, since they can be elected in the second phase to ensure the $k$-fold dominating set property. The second election process is also a random election, and the attacker cannot predict its result before the disclosure of the selected IDs. However, an active attacker can influence the election process by jamming or sending false messages and assess with different probabilities which nodes will become cluster heads.

4.3.8 Efficient computation of maximal independent sets in unstructured multi-hop radio networks

**Description:** In [112], the authors propose a clustering algorithm that works under a model capturing the characteristics of the initialization phase of unstructured radio networks. The algorithm computes a Maximal Independent Set (MIS) whose elements will be the cluster heads of the network. It is assumed that nodes have access to three independent communication channels, such as $\tau_1$, $\tau_2$ and $\tau_3$ and they can use them simultaneously. The network is modelled as a unit disk graph and time is divided into time slots. In the absence of collision detection, a node receives a message on a $\tau_i$ channel in a time slot only if exactly one node used that channel in the given time slot.

The algorithm is divided into three parts. The first part is the main loop, which selects some nodes as candidates for being cluster heads. The second part is the candidacy phase, which selects the cluster heads from the candidates. The third part is the receive triggers part, which is executed simultaneously with the other two. It handles the reception of messages on each communication channel.

A node starts executing the main loop upon waking up. First, it waits for messages on all channels in order to avoid interference with the already competing nodes. If no messages are received the node starts a loop, in which it becomes a candidate with probability $p$. $p$ starts from a very small value, and it is doubled in every iteration of the loop. If the node becomes a candidate, it broadcasts a message on $\tau_1$ channel in a time slot only if exactly one node used that channel in the given time slot.

In the candidacy phase, a candidate sends in each time slot on $\tau_2$ with probability $q$. After sending the first time, the node becomes excited and starts increasing a counter in every time slot, which is attached to each message. Upon receiving a message on $\tau_2$ by another candidate, the receiver compares the sender’s counter with its own. In case its counter is less than $8\log n$ * the sender’s counter (where $n$ denotes the number of nodes in the network) it resets its own counter. This prevents two neighboring nodes to join the MIS. When a node’s counter reaches a given value, it sends a message on $\tau_3$. The receivers of this message stop executing the clustering process and change their status covered.

**Analysis:** The termination property is satisfied only assuming a passive attacker (in the absence of interference, the algorithm terminates in polylogarithmic time). If an active attacker replays the first message sent on $\tau_1$, only one node, the sender of the first message, can change its status to candidate, other nodes have to restart the main loop continuously while receiving the replayed message.

The functionality property is satisfied in case of a passive attacker. Nodes in the MIS form the set of cluster heads, while other nodes are the members of the neighboring MIS node. Since the
termination property is not satisfied in case of an active attacker, neither the functionality property can be met. Moreover, even if we assume that the protocol terminates, the protocol would not be functional because inconsistency can occur. If an attacker jams a message sent by \( v \) on \( \tau_3 \), \( v \) considers itself as a MIS node, while its neighbors continue competing for becoming a MIS node. Later, if one of them, let’s say node \( w \), sends a message on \( \tau_3 \), the attacker can jam also this message. Both \( v \) and \( w \) consider themselves as a MIS node, while their common neighbors consider only \( w \) as part of the MIS.

Non-manipulability is also satisfied assuming a passive attacker. If the attacker cannot interfere, it is not able to manipulate roles and associations. On the other hand, active attackers can influence the election process as we discussed above.

Unpredictability is also satisfied in case of a passive attacker. In each phase, every node changes its status based on a randomly selected value \( p \). Since the attacker does not know, which nodes are awake and which are still sleeping in the main loop, every node has the same probability to become candidate from its point of view. The candidacy phase is also based on a randomly selected value \( q \). Moreover, every node resets its own counter if receives a message on \( \tau_2 \) with a value close to its own counter. Therefore, the attacker cannot predict the cluster heads before they announce it. On the other hand, an active attacker can influence the cluster head election by jamming or sending false messages and assess with different probabilities which nodes will become cluster heads.

4.3.9 LEACH

Description: LEACH (Low-Energy Adaptive Clustering Hierarchy) [61] is one of the most popular clustering algorithms for wireless sensor networks that utilizes randomized rotation of local cluster base stations (cluster heads) to evenly distribute the energy load among the sensors in the network. It creates the clusters by running a distributed algorithm, where the nodes make autonomous decisions without any centralized control, and measuring the received signal strength of the links to the neighboring nodes. In LEACH, all the data processing such as data fusion and aggregation are local to the cluster and the cluster head nodes act as gateways to the central base station.

The operation of LEACH is broken up into rounds thus requires synchronization among the nodes. Each round begins with a set-up phase, when the clusters are formed, followed by a steady-state phase, when data transfers to the base station occur.

In the set-up phase, each node decides with a certain probability to become a cluster head for the current round. This probability is based on a predetermined ratio of cluster heads for the network and the number of times the node has been a cluster head so far. Hence, to make the decision every node chooses a random number between 0 and 1. A node \( i \) becomes a cluster head for the current round if this number is less than a threshold \( T(i) \) which is set as:

\[
T(i) = \begin{cases} 
1 - p \times \left( r \mod \frac{1}{p} \right) & \text{if } i \in G, \\
0 & \text{otherwise}
\end{cases}
\]

where \( p \) is the desired percentage of cluster head nodes in the network, \( r \) is the current round number, and \( G \) is the set of nodes that have not been cluster heads in the last \( 1/p \) rounds. Computing the threshold in such a way allows each node to become a cluster head at some point within \( 1/p \) rounds. During round 0, each node can become a cluster head with the same probability (\( p \)). However, if a node elected itself cluster head in round 0 it cannot be cluster head again for the next \( 1/p \) rounds. After \( 1/p - 1 \) rounds \( T = 1 \) for any nodes that have not yet been cluster heads, and after \( 1/p \) rounds all nodes are once again eligible to become cluster heads.

If a node elected itself a cluster head for the current round it broadcasts an advertisement message to the rest of the network using a predefined transmit energy. Hearing the advertisements each non cluster head node selects the cluster to which it will belong for the given round. This selection is based on the received signal strength of the advertisement messages. Thus the cluster head node will be selected whose advertisement was heard with the largest signal strength because this cluster head can be reached using the least communication energy. After the selection each non cluster head node informs the selected cluster head about its cluster membership.
In the steady-state phase, first the cluster head node based on the number of nodes in its cluster creates a TDMA schedule telling each node when it can transmit, and broadcasts this schedule back to the cluster members. When this schedule is fixed, data transmission can take place and the nodes send their data to the cluster head using their allocated time slot. Upon receiving all the data from the cluster members the cluster head node can compress and combine the received data into a single signal and send it to the base station. When data transmission is over in all the clusters the next round begins with running again the cluster head election procedure.

**Analysis:** LEACH satisfies the termination property, because the operation is broken up into rounds. Each round consists of two phases whose terminations are ensured by using time bounds. Even an active attacker cannot prevent the nodes from terminating the different phases because the nodes decide when the given phase ends using local timers. Hence, it is also guaranteed that by the end of the set-up phase each node either considers itself cluster head or cluster member depending on the selected random value.

LEACH satisfies the functionality property but only in case of a passive attacker. As the cluster head election is based solely on local information, passive but even active attackers cannot influence the election process. Moreover, the decision about the node’s role is guaranteed by the end of the set-up phase as we discussed above. However, an active attacker can bring the system into an inconsistent state by, for instance, sending a fake cluster head advertisement message with high transmit energy. In this case, the closest non cluster head nodes will select the attacker node as their cluster head which prevents the appropriate working of this cluster for the given round.

LEACH guarantees non-manipulability assuming a passive attacker because such an attacker cannot influence the execution of the set-up phase. At the same time, an active attacker can always send fake messages about its role or in the name of other nodes due to the lack of authentication which can manipulate the outcome of the protocol.

Similarly, LEACH ensures unpredictability in case of a passive attacker because the cluster head election process is probabilistic and uses only local information. This doesn’t hold for an active attacker as such an attacker can influence the execution of the set-up phase and thus predict cluster membership to a certain extent.

LEACH, like all the other investigated protocols, doesn’t show any protection against identifiability. Even a passive attacker can learn which nodes are cluster heads because they advertise their role in the network without using any cryptographic mechanism. Similarly, knowing the positions of the nodes and thus the distances between them cluster membership can be easily identified.

### 4.4 Summary and outlook

In this section, we defined the taxonomy of dependable cluster head election and clustering, we identified the properties of dependability, such as termination, functionality, role non-manipulability, association non-manipulability, role unpredictability, association unpredictability, role un-identifiability and association un-identifiability, and we defined three adversary models, such as passive, active and compromising, that cover the possible failure and attack scenarios. Then, we described a set of relevant state-of-the-art cluster head election protocols, and systematically analyzed them with respect to the defined dependability properties in the different adversary models. We showed that all the protocols have weak dependability: assuming an active or compromising adversary none of the defined dependability properties but termination is guaranteed by the analyzed protocols.

Our future work regarding dependable clustering and cluster head election consists in the design of new cluster head election protocols that show increased dependability properties. In particular, we are currently working on a protocol that could achieve both unpredictability and unidentifiability (role as well as associations) in the strongest attacker model (compromising adversary). We intend to report more on this work in upcoming publications and project deliverables.
5 Dependability of MAC protocols

In general terms, dependability at the MAC (Medium Access Control) layer refers to reliability and security of the MAC protocol used in the WSN. In this section, we first give an overview of the relevant notions of reliability and security in this context, and then we review and analyze selected MAC protocols proposed in the literature with respect to their dependability properties.

5.1 Notions of reliability

The reliability of a MAC protocol can be defined and measured as the probability that there exist an operational link between the nodes within the transmission range. Reliability computations can be \(k\)-terminal [28] or all-terminal [126] reliability. \(k\)-terminal reliability is used to calculate the probability of connectivity for at least \(k\) nodes, while all-terminal reliability refers to the calculation of the reliability for the whole network.

For the MAC layer we consider node-to-node reliability. In some cases, depending on the protocol, reliability can be measured for two-hop neighbors.

For the MAC layer we are concerned with the existence of bidirectional links between the nodes within transmission range. With each link is associated a probability \(p_a\) for successful transmission of the packet. Further, each link has a probability \(p_{BER}\) (where BER stands for Bit Error Rate) of a bit being misinterpreted due to noise. Probability \(p_{BER}\) is a measurement of the physical properties of the environment. Therefore the link reliability is: \(p_a \times (1 - p_{BER})\) and the reliability of the node \(x\) is

\[
R(x) = \prod_{i=1}^{l} p_{a,i} \times (1 - p_{BER,i})
\]

where \(l\) is the number of links from the node.

In order to increase \(p_a\) and, therefore, to increase the reliability on MAC layer we need to increase the probability for the node to successfully access the medium. This probability jointly depends on specific mechanisms for medium access deployed by the particular MAC protocol and traffic picture in the communication range for this node. Below we introduce MAC primitives used either individually or in combination with other primitives in virtually all MAC protocols.

Reliability primitives: Now, we describe selected techniques for implementing reliable communication at the MAC layer:

Time and frequency schedules Conflict free protocols divide time or radio bandwidth into slots. Each node is assigned either one or a sequence of slots for sending and (or) receiving the data. Such allocation plan is called a transmission schedule. In conflict free protocols there no packet loses due to collisions. Scheduling protocols like TDMA, FDMA or their variants are considered as conflict-free, thus the probability of successful packet transmission is 1. A problem of finding an optimal schedule for a general class of topologies is NP-complete [125]. The solution for NP-complete problems can be approximated by various techniques. Considering wireless sensor nodes with limited computing and communication resources many of such solutions are not practically useful. An obvious drawback of this reliability primitive is inefficient usage of bandwidth under lightly loaded traffic conditions. Other drawbacks include potentially large transmission overhead due to clock synchronization, establishing and updating the schedules.

Request-To-Send / Clear-To-Send This primitive involves an exchange of short control messages prior to the actual data transmission. A source node sends a short Request-To-Send message (RTS) to the intended receiver. The receiver, if awoken, will answer with a Clear-To-Send (CTS). The nodes which overhear the RTS/CTS exchange nodes will defer from transmission until the communication between the two nodes ends. The collision can also occur during the RTS/CTS exchange.
**Virtual Carrier Sense**  A virtual carrier sense is implemented in nodes by building up a table from overheard RTS/CTS message exchange. The medium is considered free if there are no transmission indications in the table.

**Physical Carrier Sense**  A physical carrier sense is achieved by detecting the energy level in the medium. The medium is considered free if the energy level is above the pre-defined threshold.

**Preambles**  Preamble is often a short message or a stream of bits, indicating that the transmitting node has a message to send. The length and the content of this message depend on the specific protocol.

### 5.2 Notions of security

In this subsection we describe some possible attacks and corresponding countermeasures at the MAC layer [47, 57, 94, 161, 168].

**Authenticity Impersonation - Secure Authentication**  The intruder pretends being a member of the network and injects malicious data into the network. Minimizing success of this attack in order to minimize the success of this attack message authentication techniques should be implemented at MAC layer [94].

**Eavesdropping Message - Encryption**  The message is eavesdropped by an unauthorized node. To prevent the unauthorized data access the message must be encrypted.

**Compromised Integrity - Signing**  The attacker modifies the data during transmission. By signing the packets a receiver can verify the content of the message. It has been suggested that using Encrypt-Then-Authenticate techniques is the most secure procedure [94].

**Denial of Service**  DoS attacks in WSN can be achieved by jamming the radio channel [161, 168]. If the jamming is done with high power, there is no technique to prevent it. However, the edge nodes might, for example, detect lost connections and the system will be able to recognize the network failure. In further description of jamming attacks we assume that the attacker wants to stay undetected and therefore uses more sophisticated techniques to disturb the radio channel.

**Interrupt Jamming - Frame Masking**  This attack technique uses the interrupt generated upon reception of Start Frame Delimiter (SFD). When an attacker wakes up after receiving the SFD sequence on the physical layer it can send a jamming packet to interfere with the legitimate packet. The jamming period can be equal to the duration of a typical packet in the network. It can also be efficient in a shorter period just enough to overcome encoding redundancy and corrupt the checksum. The proposed countermeasure for this attack is *SFD frame masking* [168]. In this defense mechanism nodes agree on a pairwise shared key. This key is used to generate pseudo-random SFD sequence.

**Activity Jamming - Channel Hopping**  An attacker samples the Received Signal Strength Indication (RSSI) or performs the Clear Channel Assessment (CCA). If the values are above the predefined threshold, the attacker will initiate jamming with a full or partial transmission. This attack is energy costly for the attacker, since detecting the RSSI level or performing CCA has to be done frequently. Moreover, the measured values can exceed threshold even because of the background noise. The defense proposed for this attack is *channel hopping* [168]. Nodes use their shared key to create a channel schedule, which generates pseudo-random chipping sequence.
Scan Jamming - Packet Fragmentation  With channel hopping enabled, the attacker has to switch between channels and scan them for activity. Given 16 channel in 802.15.4, the attacker has to be able scan all the channels during the time period needed to transmit one packet. To defeat this attack packet fragmentation can be used [168]. In this approach each node fragments the application level data into smaller packets. These packets are then transmitted over different channels.

Pulse Jamming - Redundant Encoding  By sending a set of small packets in the air periodically, the attacker hopes to jam an existing packet in the air. The frequency of the pulse has to be high enough to disturb even small packets. In [168] the proposed defense is achieved by the redundant encoding of the header, payload and the Frame Check Sequence (FCS).

5.3 Analysis of selected MAC protocols

In this section we give an overview of selected MAC protocols for the sensor network focusing on their reliability and security properties. The interested readers can discover a more detailed information, motivation and the test results of protocols in the referred papers.

5.3.1 S-MAC

S-MAC [179] reduces the listening time using periodic sleep schedule (a duty cycle). During sleep state the node turns its radio off. A timer interrupt is scheduled according to the duty cycle. When it fires, the node wakes up and turns the radio on. The listen interval is fixed, bases on the physical layer and MAC-layer parameters, like contentions window size. All nodes are free to choose their own sleeping schedule. To enable communication, the nodes exchange their sleeping schedule with their immediate neighbors by periodically broadcasting synchronization, (SYNC), packets. The SYNC packet contains the address of the sender node and its sleep time. Sleep time is relative to the moment that sender starts transmission. This technique prevents long-term clock drift.

S-MAC includes virtual and physical carrier sense and the RTS/CTS exchange. The VCS is done by looking up local table with sleep schedules of the neighbors. The medium is determined to be free if both VCS and Physical Carrier Sense, (PCS), indicate that it is free. The node divides the listening period in two parts, the first one for the SYNC packet and following one for data packets. Each part has a contention window with many time slots for sender to perform carrier sensing. During the SYNC time period collisions can occur and the node can miss the exchange of its neighbors’ schedules. Hence the node will perform neighbor discovery with the frequency depending on the number of neighbors it has.

S-MAC also proposes an adaptive listening technique. The basic idea is to let node who overhears its neighbor’s transmissions to wake up a short time at the end of the transmission. Subsequently it would improve the latency because the node is already awake and ready to retransmit if necessary.

5.3.2 T-MAC

T-MAC [178] is based upon the S-MAC protocol but introduces an adaptive duty cycle by dynamically ending the listen period when no activation event has occurred for a time threshold $TA$. The basic idea of the T-MAC protocol is to reduce idle listening by transmitting all messages in a burst. Nodes communicates with a RTS/CTS/DATA/ACK scheme, which provides reliable transmissions and collision avoidance.

Activation events in the T-MAC protocol are: The firing of a periodic frame timer; the reception of any data on the radio; the end-of-transmission of a nodes own data packet or ACK; the knowledge from the overhearing; the neighbor ending the data exchange. The T-MAC protocol moves all communications to the beginning of the frame, thus messages are buffered between active times. An upper bound for the frame delay is determined by the capacity of the buffer.

The contention for the medium happens normally at the beginning of the active period, because messages are transmitted in bursts at the beginning of the frame. Therefore, increasing the contention window is not useful in this case. Hence T-MAC starts the RTS transmission by waiting and listening
for a random time within a fixed Contention Interval). If node does not receive CTS it retransmits RTS two more times and if no CTS was received node goes to sleep.

To avoid early sleeping problem the lower limit on the length of $TA$ is calculated as

$$TA > length(CI) + length(RTS) + T_{\text{turn-\text{around}}}$$

Overhearing avoidance is similar to S-MAC but in T-MAC it is optional.

### 5.3.3 DS-MAC

DS-MAC [96] takes its foundation in S-MAC, thus if not otherwise written, DS-MAC behaves as S-MAC [179]. The protocol is motivated by the question ”to sleep or not to sleep?”. DS-MAC implements a dynamically changing sleeping interval with fixed duration listen period. Therefore the duty-cycle is adjusted to meet the changing traffic requirements.

To deal with the clock drift DS-MAC organizes nodes into groups of peers which exchange SYNC packets. Each node maintains a synchronization table for its neighbors. Whenever DS-MAC hears a new SYNC packet the timer will be updated accordingly to the one in the SYNC originator. The SYNC packets additionally contain a new field indicating sender nodes’ duty cycle.

Each receiver node keeps track of its energy consumption, efficiency and average latency. From the start all nodes have the same duty cycle, but when latency becomes intolerable the node shortens its sleep period by half, without changing the listening period, doubles duty cycle. The receiver of SYNC packet checks the queue for packets to SYNC initiator.

### 5.3.4 PMAC

PMAC [188] is a protocol that adaptively determines the sleep-awake schedules for a node. It is based on its own traffic and the traffic pattern of its neighbors. Using the patterns received from its neighbors, the node decides when to enter a long sleep period. The sleep-wake up pattern indicates the intended sleep-wake up plan. The pattern is represented as a binary string where 0 represents the intention to sleep and 1 the intention to be awoken during the time slot. Sleep-wake up schedule is a string of bits representing the actual sleep-wake up plan. Generation of schedules and patterns imitates the slow-start algorithm used in TCP [84].

The pattern exchange is done as follows: the time is divided into frames called Super Time Frames (STF). The STF is also divided into frames which are called Pattern Repeat Time Frame (PRTF) and Pattern Exchange Time Frame (PETF). During PRTF, the node repeats its current pattern, this is divided into $N$ time slots with duration $T_R$ each, plus additional time slot $w$ during which all nodes are awake. This time slot $w$ is used to prevent long delays when activity along the nodes is low and they have long sleep patterns, or for broadcasting. PETF is divided into time slots of duration $T_E$. The last generated patterns becomes the pattern used during next PRTF and will be advised during PETF. $T_R$ is set to be long enough to handle contention window plus RTS/CTS/DATA/ACK transmission. $N$ depends on the application, the number of $T_E$ in PETF is set to the maximum number of neighbors the node could have and the span of it is long enough to broadcast a pattern.

The actual sleep-wake up schedule is generated using the knowledge of traffic patterns of node neighbors and the state of the buffer, i.e. whether there is a packet to transmit or not.

### 5.3.5 TRAMA

TRAMA [124] is a TDMA based algorithm. Time is divided into random-access and scheduled access. In random-access it establishes two-hop topology information. Channel access is contention based. TRAMA assumes that MAC can calculate the needed transmission duration based on the information from the application layer. During the scheduled access, the node announces the slots it will use as well as intended receivers. Intended receivers are calculated from bit-maps corresponding to two-hop neighbors. Priority is calculated with hash functions and slot identities.
5.3.6 FLAMA

FLAMA [123] is a scheduling based MAC protocol. FLAMA uses the predictability of traffic flows in the networks. It gathers traffic information by allowing the application to specify the traffic characteristics, or uses traffic predictions in each node. Further, FLAMA uses that information when determining schedules, and time when nodes should receive or turn radio off. The three main features of the FLAMA protocol are as follows:

1. Based on two-hop neighborhood information and implicit traffic information, the node maintains a distributed schedule which is energy-efficient and collision-free;
2. The communications between nodes with limiting processing and storage requirement performs with low transmission delay;
3. The protocol is robust even with changing topology.

5.3.7 WiseMAC

WiseMAC [42] is a one channel non-persistent CSMA with preamble MAC protocol. The listen periods are the same for all nodes, the sleep schedules are node-specific and independent from other nodes’ schedules. If a node wakes up and the medium is busy it will listen until the packet is received or medium becomes idle. Initially the preamble size is equal to the sampling period. WiseMAC does not use any handshaking technique. WiseMAC dynamically decides the length of the preamble. The decision is based upon knowledge of the sleep schedules for the transmitting neighbors. Nodes learn and refresh neighbors sleep schedule during every data exchange. This information is stored in a table. Transmissions are scheduled so that the destination nodes sampling time equals half of senders preamble.

5.3.8 B-MAC

B-MAC [122] is built upon CSMA with preamble. When a node has a message to send, it first transmits a long preamble followed by the message. The time duration when the preamble is transmitted is long enough to cover the time nodes is sleeping. When the node hears the preamble it will wait until its end and then receive the message. Then the message reception is acknowledged by sending an ACK message.

5.3.9 X-MAC

X-MAC [15] is built upon a class of asynchronous duty-cycled MAC protocols. It is an improvement of the B-MAC protocol. The protocol is designed to address the number of problems in order to improve performance and reduce energy waste of the MAC protocol. The following problems are particularly addressed: overhearing, excessive preamble and incompatibility with packetized radios.

X-MAC tackles the overhearing problem by dividing the usual long preamble packet into a series of short ones and including the targeted receiver ID. During the synchronization stage the sender transmits a short preamble packet and listens for a short period of time for the acknowledgment. The period during which the transmitting node sends a preamble packet has to be greater than the maximum length of the receivers sleep period.

When the receiving node wakes up and receives the preamble packet, it checks for an ID. If the ID does not match the node’s ID it immediately returns to sleep, otherwise the node will send an early acknowledgment (EACK) and stays awoken to receive the subsequent data packet. After the reception of the intended data packet the node stays awoken for the period of time equal to the maximum duration of back-off time in case there is another node wanting to transmit.

If the transmitter is about to send and detects the ongoing preamble, the node will listen to the channel for the EACK. If the node overhears the EACK with the intended receiver ID, then the node will back-off for a random amount of time and transmit data without preamble. Otherwise, it will start with the preamble. The back-off time has to be long enough to allow the initial transmitter to complete its data transmission.
5.3.10 MFP-MAC

MFP-MAC [7] stands for Micro-Frame Preamble MAC. To avoid irrelevant data reception MFP-MAC replaces continuous preamble by a series of small frames. Each frame contains an indicator of the data frame content, such as destination address and a hash in the data field. This allows the node to estimate the time needed for the actual data transmission and sleep until then. The node can switch the radio off if it finds out that the data to be transmitted is irrelevant. The node is able to estimate the time when the whole transmission ends and wakes up only at the end.

The frames are numbered from 1 to \( m \), where \( m \) is the amount of micro-frames needed so that their transmission lasts at most the check interval \( T_w \). This number is included in the Micro-frame. When the nodes share the same check interval, \( m \) is the same for all nodes. Hence when a micro-frame is received the node can estimate the time for the arrival of the data and for how long it can sleep. Nodes keep a table with hash-data, sent in the micro-frame, therefore minimizing redundant reception. The authors eliminates possibilities of hash collisions because of the fact that the table is cleared after each timeout. The authors do not expect a lot of simultaneously active broadcasts and expect that a hash function has good properties.

5.3.11 DEEJAM

The DEEJAM MAC protocol [168] is based upon 802.15.4 [2] hardware layer. DEEJAM depends on TinySeRSync [147] time synchronization routine. The protocol combines defense mechanisms for interrupt, activity, scan and pulse jamming attacks by using frame masking, channel hopping, packet fragmentation and redundant encoding respectively. Using these techniques all together introduces overhead, in some cases 190\%. However implementing these techniques the network performance degraded by 11\%. Therefore, the overhead is justifiable. DEEJAM was evaluated in a node to node communication scenario with one attacker.

5.3.12 Dragon-MAC

Dragon-MAC [94] implements a protocol for point-to-point encryption, using symmetric encryption with the Dragon stream cipher. The Dragon-cipher is faster than any other RC4 software implementations. Dragon-MAC result in 6 byte overhead, where 2 bytes are control bytes and 4 bytes for MAC authentication. To achieve best security it is suggested to use Encrypt-then-MAC, this will provide both privacy (encrypt) and authentication (MAC).

5.4 Summary and outlook

Table 4 gives a summary of the dependability primitives in the protocols reviewed in this section. As it can be seen, there is no MAC protocol that would fulfill dependability aspects with respect to both reliability and security. For the usage of WSN in applications connected to protection of critical infrastructures the MAC protocol should be both implementing the reliability and security mechanism. The overhead in terms of bandwidth and energy consumption of such combination should be investigated.

5.5 Future Work

In this section we considered major representatives of MAC protocols for wireless sensor networks. An obvious question to pose in this context is which of the protocols the developer should choose to fulfill the performance requirements of the CIP application?

This question would be easy to answer given an established ranking system of different protocols with respect to their contribution to the overall system performance. A rather straightforward way of constructing such a ranking system is to compare performances of different protocols in different communication contexts. Most of the existing results reported in the literature are produced using this approach. The major shortcoming of such analysis is that in some specific scenarios while a particular protocol would demonstrate better performance, this performance is still not optimal for the considered
Table 4: Dependability properties of MAC protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Reliability primitives</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-MAC</td>
<td>RTS/CTS, virtual and physical carrier sense.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>T-MAC</td>
<td>RTS/CTS, virtual and physical carrier sense. Adaptive duty cycle.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>DC-MAC</td>
<td>RTS/CTS, virtual and physical carrier sense. Dynamic duty cycle.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>PMAC</td>
<td>Sleep scheduling, RTS/CTS.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>TRAMA</td>
<td>TDMA, scheduling.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>FLAMA</td>
<td>Scheduling based on traffic.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>wiseMAC</td>
<td>Random access, CSMA with preamble.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>B-MAC</td>
<td>CSMA with preamble.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>X-MAC</td>
<td>CSMA with preamble.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>MFP-MAC</td>
<td>Scheduling with preamble, CSMA.</td>
<td>Not considered.</td>
</tr>
<tr>
<td>DEEJAM</td>
<td>Scheduling, random access.</td>
<td>Jamming resistance.</td>
</tr>
<tr>
<td>Dragon-MAC</td>
<td>Not considered</td>
<td>Authenticity, integrity, confidentiality.</td>
</tr>
</tbody>
</table>

operating environment. This is because it is not a particular protocol as an inseparable unit which in cooperation with other inseparable protocols produce one or another performance picture. In this section we, for example demonstrated that none of the MAC protocols may fulfil both reliability and security requirements of the application. It is the specific protocol’s component (which may be present in several other protocols) and its particular parametrization for a given scenario that produces one or another impact on the overall system performance. An optimal combination of the components for a specific scenario may not be present in any existing monolithic protocol, and therefore will constitute a completely new protocol.

Being able to combine functional components into an application-specific protocol (or in other words having a modularized structure of MAC protocols) is the essential prerequisite towards optimization of WSN performance. As a matter of fact modularized MAC protocols do not exist presently. In the scope of the future work within work package WP3 we will consider the specification of such modular MAC protocol along with the framework for verification of the dependability properties of the combination.
6 Dependability at the physical layer

Wireless sensor networks can be deployed in operational environments where not only their confidentiality and integrity is important, but also their availability. In particular, this is the case in Critical Infrastructure Protection applications. In such applications, the availability and the resistance against Denial-of-Service attacks of the WSAN is indispensable.

However, wireless nodes are usually equipped only with low power radios and their transmission power is very limited. This opens up the possibility for an attacker to jam the network nodes and render the network non-functional.

Jamming can occur as a result of an attack or can be caused by other devices that unintentionally interfere with communication. Intentional or not, jamming is a persistent threat to the reliability and availability of wireless networks. Spread spectrum techniques are usually employed to cope with jamming, where jammed devices either switch sequentially to different transmission channels or spread the signal in time or frequency to reduce the probability of jamming. However, current sensors, and most consumer devices, do not have sufficiently sophisticated radio devices to implement these techniques effectively. Also, the difficulty of having a network wise secret to seed pseudo random sequences makes the adoption of these technique problematic.

While anti-jamming techniques play an important role in ensuring availability, a jammer can always succeed if she is prepared to emit radio signals at the necessary power. Hence, there is a limit to what anti-jamming can achieve. This is particularly true for wireless sensor networks, where nodes are typically battery power and transmit at extremely low power.

In this section, we first define models and metrics for jamming attacks that can be used to classify them and measure their effectiveness, respectively. Then, we give a brief overview of known anti-jamming techniques, including techniques based on the principles of spread spectrum communications, and other techniques specifically developed for wireless sensor networks. Finally, we sketch what we intend to achieve in the WSAN4CIP project with respect to jamming protection, and availability in general.

6.1 Jamming attack model and metrics

6.1.1 Adversarial model

There exist many different attack strategies for jamming. We follow the classification for jammers devised in [170]:

- **Constant Jammer**: The constant jammer continuously emits a radio signal, possibly on all available channels. This jammer is the most effective but also easy to localize and less energy efficient.

- **Random Jammer**: This kind of jammer only emits according to a certain jamming probability. When this probability is 1 then it is a constant jammer. This kind of jammer can be particularly effective if one considers that it might be sufficient to corrupt a small portion of the messages in order to prevent them to be correctly received.

- **Reactive Jammer**: This is a form of active jammer makes sure that the parties are indeed communicating before emitting a jamming signal. The jammer remains idle if there is no transmission, and starts emitting a radio signal as soon as it senses activity on the channel.

- **Smart Jammer**: This kind of jammer uses her knowledge of all the protocols in the network stack (data link, routing, etc.) in order to increase the effectiveness of the attack. While this attacker is still a jammer, she can target specific messages types, message parts or nodes in the network.

Regarding this last type of attacker, we note that, for example, most data link layer protocol perform poorly against DoS attacks [166]. [114] analyses the impact on transmission when the attacker specifically targets request to send (RTS) packets in a CSMA/CA MAC protocol. While a purely
physically jammer can corrupt a portion of the traffic that is proportional to the power spent, the
attack strategy in this work shows a cascade effect on the MAC layer that significantly increases the
effectiveness and power efficiency of the attack.

6.1.2 Success metrics
When discussing jamming we also need means to measure when a jamming attack is successful. [170]
considers the problem for one-to-one communication in the presence of a jammer. The two metrics
devised are:

- Packet Send Ratio (PSR): which is the ratio of packets correctly sent over the packets that were
  scheduled for sending at the data link layer. This is because most data link protocols use some
  form of carrier sensing technique, so that if the channel is busy the transmission is delayed;
- Packet Delivery Ratio (PDR): this is the ratio of packets correctly received over the ones that
  are sent.

This metrics, only consider a one-to-one communication pattern and are therefore limited. It is
still an open research challenge, to measure the network wide impact of jamming in an ad-hoc setting.

6.1.3 Jamming attacks on sensor networks
In [166] the authors present various attacks to 802.15.4 sensor nodes aimed at showing that even
a single mote class device is able to perpetrate considerable damage. In their attack they employ a
MicaZ node that acts as a reactive smart jammer. By only jamming specific parts of the MAC layer
packets and reacting when messages are sent, this jammer manages to be highly effective while saving
battery power.

6.2 Anti-jamming techniques
6.2.1 Spread spectrum techniques
Cryptography alone can not protect against jamming or other kinds of physical attacks. Well es-
established countermeasures against jamming include frequency hopping and direct spread spectrum.
These techniques are usually referred to as spread spectrum techniques, since the transmitted signal
takes up more bandwidth than the information signal that is being modulated. Spread spectrum sig-
als are more resistant to narrowband interference than fixed-frequency transmission. Furthermore,
spread spectrum transmissions are difficult to intercept and eavesdrop without knowing the hopping
sequence. Finally they provide better channel utilization and throughput even in the absence of an
attacker, that is when many devices share the same frequencies.

Frequency Hopping  Frequency-hopping spread spectrum (FHSS) is a simple technique, where the
communicating parties rapidly switch among many frequency channels, using a secret pseudorandom
sequence shared amongst them. The pseudorandom sequence is usually derived from a shared secret
key using a PRF.

Frequency hoppers are resistant to jamming by an adversary that does not know the hopping
sequence. However, if hopping is slow and the adversary can switch amongst the channels rapidly,
then following jamming is possible. 4

Direct-sequence spread spectrum  Direct-sequence spread spectrum (DSSS) is a modulation
technique used to evade jamming. As in FHSS, the signal is spread over the spectrum of a device’s
transmitting frequency. In practise, the signal is multiplied by a pseudorandom sequence to occupy
the entire spectrum. Like FHSS it makes jamming substantially harder, but it can also make the
signal harder to intercept [5].

4For example, the MicaZ mote has a slow channel switch time of 50ms
Uncoordinated recovery  Spread spectrum techniques rely on a shared secret to generate a pseudo-random sequence. However, Uncoordinated Frequency Hopping (UFH) [145] addresses the problem of two nodes, a sender and a receiver, that do not have a secret shared key, but still wish to communicate in the presence of a jammer. The lack of a pre-shared secret prevents the two nodes from communicating using spread spectrum techniques.

With UFH, both the sender and the receiver hop among a set of frequencies in a random and uncoordinated fashion. At each hopping phase there is a small probability that they both the sender and receiver transit on the same channel. However, this probability increases with the number of attempts and, in time, they are able to complete a key exchange even in the presence of a jammer. Once a new key is established, traditional spread spectrum techniques can be employed.

6.2.2 Other anti-jamming techniques for WSNs

Jamming defenses on WSNs are mostly derived from the ones described above. However, the limited power, lack of specialized hardware and the possibility of node compromise pose difficult research challenges that have been addressed with techniques specific to WSNs.

Spatial retreats In [172], channel surfing and spatial retreats are proposed to evade jamming attacks. Channel surfing is similar to frequency hopping, however nodes wait for a channel to be jammed before they move to the next channel in the sequence. In order to do so, they develop an algorithm for jamming detection. The second defence strategy, spatial retreats, assumes mobile nodes that, upon detection of jamming, simply move away from the jammed region. This involves detection of the jammed region and evasion, in such a way that connectivity is preserved.

Mitigating control channel jamming [148] addresses the problem of identifying colluding insiders in a WSN that are jamming control messages of the data link layer. They reduce the problem of robust control channel access in case of jamming to the problem of secure key establishment with node capture. Randomly assigned key are used by the nodes to access the control channel. An greedy algorithm is then used to identify jammers and colluding nodes inside the network.

Timing channels In [171], a novel jamming evasion strategy is proposed that involves the establishment of a timing channel between nodes, in spite of the presence of a jammer. The timing channel is a low-rate physical layer overlay on top of the link layer and functions using failed packet reception times. This channel is shown to be resilient against jamming and different constructions are explored that lead to a low-rate overlay communication channel. An analysis is also present that shows the resiliency of this channel against an attacker that specifically targets timing information.

DEEJAM DEEJAM [166] is a MAC-layer protocol for defeating power efficient, mote class jammers in 802.15.4 wireless networks. The technique relies four defensive mechanism to prevent efficient jamming: frame masking, which involves masking the preamble of messages at the physical layer; frequency hopping; error correction codes to recover from corruption of portion of a message; packet fragmentation to minimize the chances of following jamming.

LDPC Low density parity codes (LDPC) [116] have been proposed for jamming countermeasures in 802.11 wireless LANs. LDPC are shown to be suitable to prevent low energy jamming where the attacker only corrupts a portion of the messages. However, due to the short packet lengths it might not be suitable for WSNs.

Jammed area mapping In [167] Wood et al. propose a protocol that allows sensors to map an area that surrounds a jammer. This way, network applications could make decision based on the knowledge of this region, rather than having to deal with single broken links and congested nodes. The protocol described is distributed and it is shown to reach convergence on the jammed area in a short period of time.
Wormholes  Cagalj et al. [18] propose using wormholes to maintain a WSN functioning even under a jamming attack. In practice, wormholes are used to transmit alarms out of the jammed region that can then be transmitted to the network operator. Three different wormholes constructions are explored: one based on wired pairs of sensors, another based on frequency hopping and a third based on uncoordinated frequency hopping.

6.3 Summary and outlook

In this section, we first defined models and metrics for jamming attacks, and then, we gave a brief overview of known anti-jamming techniques. As part of our future efforts in the WSAN4CIP Project, we will investigate the amount of energy required by an attacker in order to jam an entire WSAN. We intentionally abstract from the jamming and anti-jamming techniques. Instead, we will consider a jammer that emits on all the frequencies available to the sensors. This is because, we are interested in finding the theoretical limits of transmission in an ad-hoc network when there is interference in the transmission.

The idea is to construct an analytical model that can be applied to many different network settings. The usefulness of such a model is in showing how easy it is to jam and render unavailable a WSN. The knowledge of this work-factor, can be used in two ways: first guide in the design of physical and upper layer protocols, as it will establish a baseline for DoS attacks. Second, it will help assess the general viability and applicability of WSAN in a context such as Critical Infrastructure Protection, where the availability of the networking infrastructure might be essential.
7 On the robustness of network topologies

While many research papers assume that, in sensor networks, the nodes are deployed in a random manner (e.g., thrown out of an aircraft), we believe that in the majority of the civilian applications, and in particular in CIP applications, the sensor nodes should be deployed systematically. This has the advantage that the position of the nodes can be optimized for maximizing sensing coverage, minimizing energy consumption, and increasing the network lifetime. Besides these basic objectives, systematic node placement can help to increase the fault tolerance of the network and its resistance to jamming attacks on the links as well as to destructive physical attacks on the nodes. In other words, the careful placement of the nodes can play an important role in increasing the dependability of the sensor network.

The placement of the nodes and their radio transmission ranges determine a network topology. In order to compare different potential node deployment strategies, one needs a metric to measure the quality of a resulting topology, where quality here means resistance to random failures and intentional attacks. The most widely used such metric in the sensor network literature is the degree of connectivity of the topology graph [34]. As failures and attacks can target both nodes and links, both the concept of vertex connectivity and that of edge connectivity are relevant. The vertex/edge connectivity of a graph is the size of its smallest vertex/edge cut, and, intuitively, the larger the connectivity is, the more dependable the topology is.

The connectivity (vertex and edge) based metrics are appealing because they are intuitive and they can be computed efficiently. Yet, we argue that they fail to capture some important aspects of the dependability problem. In particular, they do not shed light on how a $k$-connected topology fails, when more than $k$ nodes/links are destroyed. As an illustrative example, consider Figure 7. The graph on the left is 2-connected, while the graph on the right is only 1-connected. Still, when an attacker can remove two nodes from these graphs, the one on the left side fails much worse than the one on the right side, because the left hand side graph completely falls apart, while the other one remains mostly connected.

![Figure 7: The notion of $k$-connectivity fails to capture what happens with the graph when more than $k$ node is removed from the graph. The left hand side graph is 2-connected, while the right hand side graph is 1-connected. However, when an attacker can remove 2 nodes from these graphs, the left hand side graph completely falls apart, while the other one remains mostly connected.](image)

One can argue that the left hand side network was not designed to resist an attacker that can remove two nodes. However, in practice, it is very difficult to accurately estimate the capabilities and resources of an attacker, and hence to determine the right degree of connectivity to be provided. It would be more desirable to design network topologies that fail gracefully as the power of the attacker increases. An important prerequisite is to define appropriate metrics that can characterize the dependability of network topologies with respect to a continuum of attacker strengths.

In this section, we give an overview of the most important concepts that have been proposed in connection with the definition and quantification of network robustness. These concepts include vertex- and edge-connectivity, but also several others. We make a comparison of these concepts, and we also propose a novel approach to define robustness as a function of attacker strength.
7.1 Existing robustness metrics

7.1.1 $k$-connectivity

Vertex- and edge-connectivity are probably the most frequently used measures for robustness. They measure the number of vertices or edges, that have to be removed in order to disconnect a graph. The related definitions are the following:

Definition 1 (Vertex cut) A vertex cut of a connected graph $G$ is a set of vertices whose removal renders $G$ disconnected.

Definition 2 ($k$-vertex-connected) A graph $G$ is $k$-vertex-connected if it remains connected when no more than $k$ vertices are removed. Formally, $G \setminus X$ is connected for all $X \subseteq V(G)$ with $|X| \leq k$.

Definition 3 (Vertex-connectivity) The vertex-connectivity $\kappa(G)$ of a connected graph is the largest $k$ for which it is $k$-vertex-connected, or in other words, the size of its smallest vertex cut.

Definition 4 (Edge cut) An edge cut of a connected graph $G$ is a set of edges whose removal renders $G$ disconnected.

Definition 5 ($k$-edge-connected) A graph $G$ is $k$-edge-connected if it remains connected when no more than $k$ edges are removed. Formally, $G \setminus X$ is connected for all $X \subseteq E(G)$ with $|X| \leq k$.

Definition 6 (Edge-connectivity) The edge-connectivity $\lambda(G)$ of a connected graph is the largest $k$ for which it is $k$-edge-connected, or in other words, the size of its smallest edge cut.

The following theorems state some of the most fundamental properties of vertex and edge connectivity:

Theorem 1 $\kappa(G) < \lambda(G) < \delta(G)$, where $\delta(G)$ denotes the minimum degree of the vertices in $G$.

Theorem 2 Connectivity is preserved by graph homomorphisms.

Another important notion is that of local connectivity:

Definition 7 (Local connectivity) Local vertex-connectivity $\kappa(u, v)$ is the size of the smallest vertex cut disconnecting $u$ and $v$. Local edge-connectivity $\lambda(u, v)$ is the size of the smallest edge cut disconnecting $u$ and $v$.

Theorem 3 Vertex- and edge-connectivity can be computed in polynomial time by finding the minimum values of $\kappa(u, v)$ and $\lambda(u, v)$ using Menger’s theorem.

As stated above, we do not believe that connectivity of the network topology graph is a meaningful metric for measuring the robustness of the network topology. In our opinion, the problems with connectivity can be traced back to two main causes:

- Edge- or vertex-connectivity is only concerned with whether the graph is connected or not. The example in Figure 7 clearly shows that this is not sufficient, as one would also be interested in how the graph falls apart when removing its vertices or edges.

- Edge- or vertex-connectivity is only concerned with a minimum. This means, that it is only concerned with attackers, which are restricted in resources and can only remove a few vertices. However, this assumption can not be generally made. Thus, a good measure should also take into account attackers with various resources and various number of vertices/edges removed.
7.1.2 An axiomatic approach

In [29] an axiomatic approach for designing vulnerability measures of topologies is introduced. Three axioms are given, which can be considered self-evident, and which all vulnerability functions should satisfy. Two example measures are also given, but without justification for their choices.

**Definition 8 (Vulnerability function)** Let $G$ denote the set of all finite graphs. A function $v : \mathcal{G} \to [0, 1]$ function is a vulnerability function, if

- $v$ is invariant under isomorphisms,
- $v(G') \geq v(G)$ if $G$ is obtained from $G'$ by adding edges, and
- $v(G)$ is computable in time polynomial in $|V(G)|$.

In the paper, the authors propose the following two vulnerability functions, and they formally prove that they satisfy the axioms above:

$$v^*(G) = e\left(\frac{M-m+n-|E|-2+\frac{2}{n}}{n}\right)$$

where $m$ and $M$ are the minimum and maximum degrees, respectively, and

$$v^{**}(G) = e\left(\frac{\sigma+n-|E|-2+\frac{2}{n}}{n}\right)$$

where $\sigma$ is the standard deviation of the degree distribution in $G$.

Although the axioms are self-evident, no justification is given why the above described measures should be used, instead of any of the infinite number of other possible functions. For example, connectivity could be used to define a measure, as it can be easily proven that both $-\kappa(G)$ and $-\lambda(G)$ fulfill the axioms.

7.1.3 Graph toughness

Toughness is a graph connectivity measure described in [11] with several related theoretical results. It fulfills the first two of the above mentioned axioms, but it is unfortunately NP-hard to compute.

**Definition 9 (t-tough and toughness)** A graph $G$ is $t$-tough, if $|S| \geq t\omega(G - S)$ for every subset $S$ of the vertex set $V(G)$ with $\omega(G - S) > 1$, where $\omega(G')$ denotes the number of components in $G'$. The toughness of $G$, denoted by $\tau(G)$, is the maximum value of $t$ for which $G$ is $t$-tough.

Intuitively, this definition requires for a $t$-tough graph $G$ that for any subset $S$ of its vertices, if removing the vertices in $S$ renders the graph disconnected, then the number of the resulting components is upper-bounded by $|S|/t$. In other words, the graph cannot fall apart arbitrarily.

The problem of computing the toughness of a graph can be formalized as follows:

**Definition 10 (t-TOUGH problem)** INSTANCE: graph $G$; QUESTION: Is $\tau(G) \geq t$?

As said before, the $t$-TOUGH problem is NP-hard [11]:

**Theorem 4** For any positive rational number $t$, $t$-TOUGH is NP-hard.

Unfortunately, $t$-TOUGH remains NP-hard even for very special cases (e.g., it is NP-hard also for bipartite graphs). This means, that in practice we can only compute a heuristic approximation of the toughness of a graph.
### Measure

<table>
<thead>
<tr>
<th>Measure</th>
<th>Computational complexity</th>
<th>Isomorphism invariant</th>
<th>Monotonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge-connectivity</td>
<td>polynomial time</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>vertex-connectivity</td>
<td>polynomial time</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>toughness</td>
<td>NP-hard</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>strength</td>
<td>polynomial time</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 5: Comparison of existing topology robustness metrics

### 7.1.4 Graph strength

Strength is a graph connectivity measure \([32]\), which corresponds to the minimum ratio of edges removed per components created. This measure is a very promising one, as it fulfills all the above mentioned axioms (including computability), and it also satisfies our intuitive requirements on graceful degradation.

**Definition 11 (Strength)** The strength \(\sigma(G)\) of a graph \(G\) is defined as:

\[
\sigma(G) = \min \left( \frac{s(A)}{\kappa(A)} : A \subseteq E, \kappa(A) > 0 \right),
\]

where \(s(A)\) is the attack cost based on edge weights and \(\kappa(A)\) is the number of new components created by the removal of \(A\).

Intuitively, the strength is smaller if a larger number of components can be created by the removal of a set of edges with a smaller overall weight. Note that when all edge weights are 1, function \(s\) simply returns the number of edges.

An interesting aspect of this metric is that it is concerned with edge removal instead of vertex removal. This fits well to model jamming problems, as a jamming attack usually renders (directed) links non-functional instead of entire nodes.

In addition:

**Theorem 5** For any graph \(G\), there exists polynomial time algorithm to compute \(\sigma(G)\).

### 7.2 Comparison of existing metrics

In Table 5, we present a comparison of the existing measures described above. We can see that nearly every measure fulfills all of the requirements, with toughness being the only exception. However, edge- and vertex-connectivity have great disadvantages. The most important of all is that they only take into account the effort needed to disconnect the graph into two components, but not the effort needed to disconnect it into several components.

Among the presented measures, only toughness is not computable in polynomial time. This is a great disadvantage, as we usually have to give a guaranteed level of security or reliability in practice. In the case of toughness, only a heuristic approximation of robustness can be given.

We have found that, among the existing measures, strength fulfills the most of our requirements and can be considered as the best candidate for practical use.

### 7.3 Future work

In this section, we identified graph strength as a promising concept that can be used to measure the robustness of a network topology. In particular, in contrast to the most frequently used vertex- and edge-connectivity metrics, which are only concerned with the question of when a graph becomes disconnected, graph strength is also concerned with the question of how a graph falls apart as a function of the cost of an attack.

The next step will be to gain more understanding of the concept of strength and its potential applications in the context of node deployment strategies. This includes the computation of the strength of some unit disc graphs (representing potential topologies of WSNs), both artificially created...
and randomly generated. The ultimate challenge, however, is to understand what properties help to increase the strength of a graph, and how such knowledge can be exploited in designing node deployment algorithms that maximize the robustness of the resulting network topology.
References


