



Master thesis on the topic

Deep packet inspection in field busses

Degree course:	Computer Science
Handed in by:	Georg Stefan Ehrke, B.Sc.
Enrolment number:	213204159
Period:	30th October 2018 – 19th March 2019
Supervisor:	M.Sc. Johannes Goltz, Dr.-Ing. Thomas Mundt
Primary reviewer:	Prof. Dr. rer. nat. Clemens H. Cap
Secondary reviewer:	Prof. Dr. rer. nat. habil. Andreas Heuer



Acknowledgments

I would like to take this opportunity to thank those who helped me with this master thesis.

First and foremost, I would like to thank Dr. Thomas Mundt and Johannes Goltz for supervising me throughout this master thesis, providing lots of valuable feedback.

Additionally, I would like to thank Johann Bauer for publishing this theme and helping me with questions regarding \LaTeX .

I also, of course, owe very special thanks to Divya Kowshik, who proof-read this master thesis and was always at hand, when i needed help phrasing my ideas in english.

Abstract

Deep Packet Inspection is a well-known system for monitoring and controlling network traffic in IP-based systems. This master thesis introduces a novel concept for Deep Packet Inspection on field bus systems used for building automation.

Parts of the concept, namely a tool for automatically generating rules, were implemented and a method for evaluation was developed.

Kurzfassung

Deep Packet Inspection ist eine stark verbreitete Methode zur Überwachung und Kontrolle von Netzwerkverkehr im Kontext von IP-basierten Systemen. Diese Masterarbeit präsentiert ein neuartiges Konzept für Deep Packet Inspection auf Feldbussystemen, welche primär Verwendung für Gebäudeautomation finden.

Teile des Konzepts, namentlich ein Tool zur automatischen Generierung von Regeln, wurden implementiert und eine Methode zur Evaluierung des Systems vorgestellt.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Research question	4
1.3	Structure	4
2	Background and prior work	5
2.1	Building Automation Systems (BAS)	6
2.1.1	Field-busses	6
2.1.2	Security demands	6
2.1.3	Protocols	7
2.1.4	Attacks	7
2.2	KNX in detail	8
2.2.1	KNX/TP1	10
2.2.2	Topology	14
2.2.3	Addressing	17
2.2.4	Communication objects	19
2.2.5	Frames	20
2.2.6	Communication	25
2.2.7	Interworking	26
2.2.8	ETS	29
2.2.9	Official Extensions	29
2.3	Network Filters	29
2.3.1	Terminology	29
2.3.2	Packet Filtering	30
2.4	Deep Packet Inspection	32
2.5	Prior work	33
2.5.1	Unofficial KNX extensions	33
2.5.2	KNX-related research	34

CONTENTS

3	Concept	35
3.1	Extended Coupler Unit	36
3.2	Metadata on the bus	39
3.3	Data in ETS	40
3.4	Filter language	42
3.5	Matching	43
3.6	State	47
3.7	Generating rules	48
3.8	Building matchers for DPTs	54
3.9	Example	55
3.10	Attacks	58
3.11	Rule parsing	59
4	Implementation	61
4.1	Generating rules from ETS	62
4.2	Remarks on implementation of ECU	63
5	Evaluation	65
5.1	Methodology	66
5.2	Assessing Results	68
5.3	Thought experiment	69
6	Conclusion	73
7	Future work	75
	List of Figures	77
	List of Tables	79
	List of Acronyms	81
	Bibliography	83
	Appendices	
A	Disc with supplementary material	

Introduction

Contents

1.1	Motivation	2
1.2	Research question	4
1.3	Structure	4

This chapter will give an introduction to the master thesis. In the first subsection, the motivation for such a project will be elaborated, to illustrate current problems in the area of field busses and building automation systems in particular, but not limited to KNX. Following this, the consequent section will narrate the general research question and highlight particular subquestions of interest. The chapter will be concluded with a general overview of the structure of this thesis.

1.1 Motivation

Building Automation systems (BASs) have become increasingly important in the last decades. Integrating various housing functions, they serve to couple numerous services surrounding building automation. Despite just gaining recent popularity among home users, building automation has been around for decades in bigger facilities and commercial buildings. However, the topic of security has long been neglected. BASs are outside the scope of traditional IT security research.

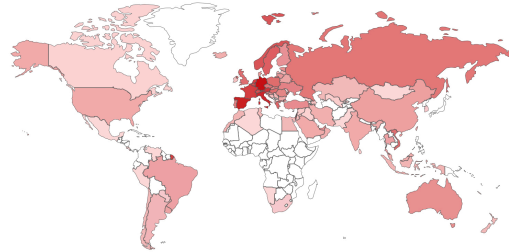


Figure 1.1: Heatmap of internet-accessible KNX Gateways

Source: Screenshot taken at <https://www.shodan.io/search?query=KNX+Gateway>

Mundt, Krüger, and Wollenberg showed in their paper “Who Refuses to Wash Hands? Privacy Issues in Modern House Installation Networks” how they could track a user’s behaviour in a building featuring a BAS by simply listening to the field bus system. [MKW12]

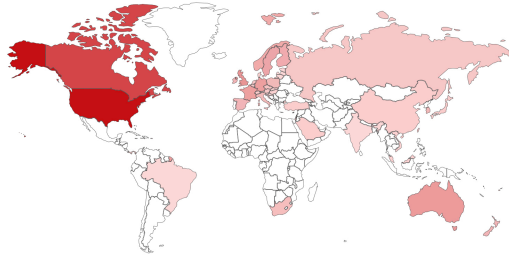


Figure 1.2: Heatmap of internet-accessible BACnet Gateways

Source: Screenshot taken at <https://www.shodan.io/search?query=%22BACnet+Broadcast+Management+Device%22>

Aside from attacks coming from within the building, BASs that are connected to the internet pose a similar threat. A small request on the internet-of-things search engine “Shodan” revealed 17,769 hits for “KNX Gateway” and 3,737 hits for “BACnet Broadcast Management Device”. A heatmap of both requests is depicted in Figure 1.1 and 1.2. Due to legal and ethical reasons, these hosts were not further analysed for vulnerabilities without prior consent from the respective administrator.

In his presentation *Securing industrial control systems - A peek into building automation security*, the security researcher Brandstetter outlined scenarios, where an attacker gains access to a building's Heating, Ventilation, Air Conditioning (HVAC) system, creates an environments that won't allow employees to work in and blackmails the companies managers. Further on, attackers with access to a building's HVAC system might cause serious harm to server hardware by simply disabling ventilation in the corresponding section of the premises.

Schwab and Poujol, authors of *The State of Industrial Cybersecurity 2018*, commissioned by Kaspersky labs, researched the state of cybersecurity. They conducted surveys with major businesses and found that: "More than half of the companies did not experience any incident or breach in the past 12 months. Although this seems to be a good thing at first glance, the question is whether or not they would even have recognized it. Many companies do not detect or even track attacks!". [SP18, p. 4] Companies that did experience issues with cyberthreats, suffered from a strong impact. They asked companies, how likely they consider themselves to be a victim of a cybersecurity incident. The results are shown in Figure 1.3. In all geographical regions, at least half of the companies considered themselves very likely or at least quite likely to become victims of a cybersecurity incident. This is a rather considerable increase compared to 2017.

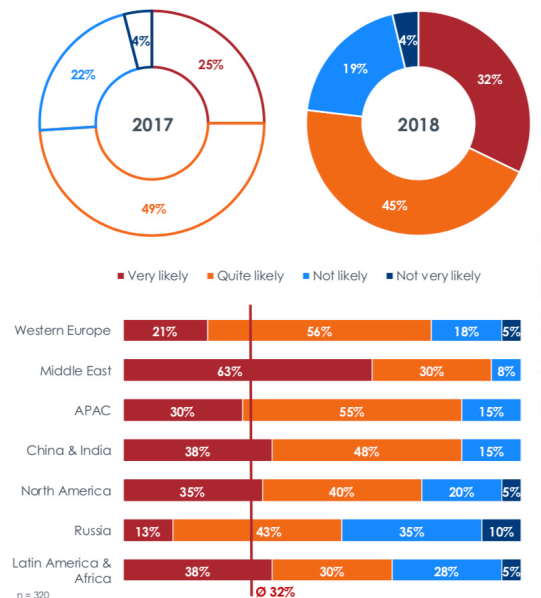


Figure 1.3: Categorisation of attacks on BAS
Source: [SP18], page 8

1.2 Research question

This master thesis will explore two major questions:

1. Is all the data essential for filtering network packets obtainable?
2. What sort of attacks can be prevented using Deep Packet Inspection?

(1) First, it must be investigated, what potential data that can be obtained. This encompasses both, data that can be extracted from the field bus system itself, and also data that can be taken from management software related to the system. Based on the findings of the analysis, it shall be evaluated whether that data is enough for building a filter system. (2) Given the filter system developed under point (1), it should be analysed and evaluated, what kind of attacks can be prevented.

1.3 Structure

The following chapter 2 “Background and prior work” will provide the necessary understanding of Building Automation systems (BASs), Network filtering for IP-based systems and concepts of Deep Packet Inspection. Further on, it will elaborate on existing solutions and research regarding securing BASs. Subsequently, chapter 3 “Concept” presents a new notion on Deep Packet Inspection for field bus systems, including an analysis of available metadata, the introduction of a filter-language, an algorithm to automatically generate a most restrictive set of filters and different approaches on sorting filters to increase performance. Following thereafter, chapter 4 “Implementation” talks about the execution of the automated tool for generating the filter set and the filtering system. Chapter 5 “Evaluation” proposes a methodology for assessing the concept. A final summary and assessment of the results is given in Chapter 6 “Conclusion”. The last chapter, chapter 7 “Future work”, presents possible further research with respect to the proposed concept and related findings.

Background and prior work

Contents

2.1	Building Automation Systems (BAS)	6
2.2	KNX in detail	8
2.3	Network Filters	29
2.4	Deep Packet Inspection	32
2.5	Prior work	33

This chapter is divided into five sections. The first section predominantly reviews BASs in general, presenting different protocols and their shortcomings. The second section explains KNX, one of the most popular protocols for BASs, in detail. Subsequently, section 3 describes network filters, followed by Section 4 dealing with Deep Packet Inspection. The last section of this chapter provides an overview of prior work.

2.1 Building Automation Systems (BAS)

Modern buildings are equipped with an increasing amount of sensors and automation. Traditional wiring, where there is a 1 to 1 connection between a sensor and an actor is no longer suitable. BASs, most commonly used with a field bus system as backbone, are on the rise. They allow the integration of systems like heating, ventilation and air conditioning, commonly referred to as HVAC, but also lighting and shading. The common backbone reduces the need to wire and enables to lower maintenance costs. But advantages of BASs are not limited to simply cost saving. They improve automation, e.g. connecting outdoor light sensors and indoor lighting or presence sensors and heating, and global monitoring / control. Besides, it also accommodates reconfiguration and additions to the buildings functionality during its lifetime. An additional benefit is monitoring, leading to early detection of malfunctions.[NG10] [GPK10] [Sok17]

2.1.1 Field-busses

Field-busses allow connections between various field devices via a common bus. Field devices can be sensors, actuators, controllers, management nodes or even gateways, to provide remote access. Each device is connected to one or more datapoints, most often representing current values. Simple switching can easily be implemented with direct wiring between sensor and load, but field-busses enable enhanced features like monitoring and automation. [Kas+05][BJD16]

2.1.2 Security demands

Wolfgang Granzer, et. al describe security demands in their publication “Security in networked building automation systems”:

- Data Confidentiality
- Data Integrity
- Data Freshness

2.1.3 Protocols

“Security in Building Automation Systems” gives an overview of different protocols for BASs and discusses their shortcomings. The four chief protocols, not only for the automation of big buildings, but also home automation for end users are BACnet, LonWorks, KNX and ZigBee. The BACnet specifications introduces measures to prevent attacks that include unauthorized interceptions, modifications of traffic. It is not considered secure, because it does not specify the initial distribution of secret keys, due to the use of Data Encryption Standard (DES) and the lack of protection against replay attacks. There exists an addendum to the BACnet specification that introduces AES and Keyed Hash Message Authentication. [GPK10] While DES is still good enough to prevent immediate Man-In-the-Middle attacks, network packets can be decrypted with a reasonable time expense, enabling an attacker to monitor user’s behaviour. LonWorks specifies a 4-step challenge-response communication, but lacks proper channels to authenticate the device of the receiver. [GPK10] KNX does not offer proper security out of the box. There is a basic access control scheme, but the keys are transmitted in clear text. To make things worse, the only configuration software, Engineering Tool Software (ETS), uses the same key for the entire installation. KNX introduced a *KNX Data Secure* extension, that will be further discussed at the end of the next section. [Gra+06] [GPK10]

2.1.4 Attacks

Despite the limitations of different implementations, general attack approaches have been the topic of research. Wolfgang Granzer et al. introduced a categorisation of attacks, notable in Figure 2.1.

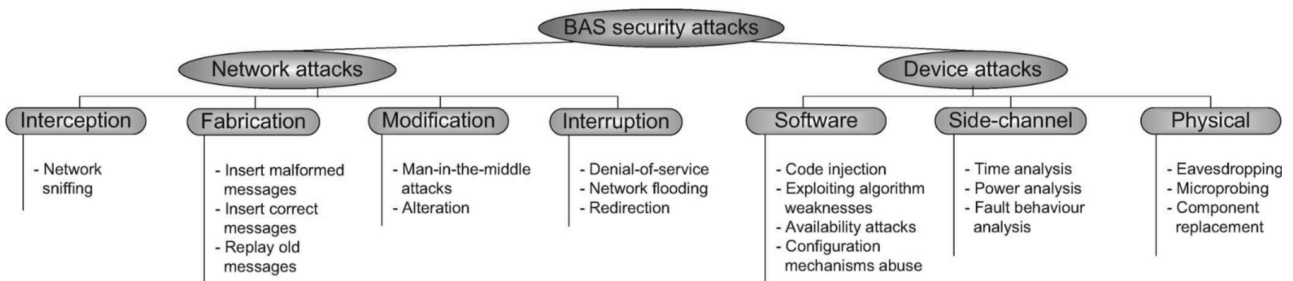


Figure 2.1: Categorisation of attacks on BAS

Source: Security in building automation systems, Wolfgang Granzer et al.

2.2 KNX in detail

The KNX Association, responsible for developing the popular home- and facility automation KNX, emerged through the merger between the European Installation Bus Association (EIBA), BatiBUS Club International (BCI) and the European Home Systems Association (EHSA) in 1999. While European Installation Bus (EIB) was mostly concerned with lighting and shading, BatiBUS dealt with HVAC and European Home Systems (EHS) with home and audio/video appliances. Among others, it is standardised in EN 50090¹. Similar to other BAS protocols, KNX is a decentralised field-bus protocol. It interconnects different devices like sensors, actuators, controllers, monitoring devices etc via a common bus system. Special field-bus devices are system components like power supply units or bridges respectively couplers, which will be discussed further in the section 2.2.2 “Topology”. Every field-bus device contains its own microprocessor. To achieve interoperability, the KNX standard offers default encoding and data schemes. This allows end-users to connect sensors and actuators of different vendors. The application model is explained further in subsection 2.2.7 “Interworking”. [Hüb18] [Sok17] [Kas+05] [Ass13]

Modes

KNX offers varied modes of configuration. The automatic mode, A-Mode, offers a plug-and-play like experience and requires no professional configuration. According to “Implementation of the KNX Standard,” there are currently no devices offering A-Mode available on the market. Easy mode, E-Mode in short, only provides a small subset of the KNX functionality. There are also fewer devices available, capable of running in E-Mode. Configuration requires neither a skilled worker, nor the use of the ETS software. Last but not least is the system mode, S-Mode. Requiring skilled workers, it allows for complex configuration and big installations. To use the S-Mode, it is necessary to configure the system using the ETS software. [Sok17][Ass13][KW06]

The S-Mode is the most popular mode of configuration and will invariably be referred to onwards, unless explicitly stated otherwise.

¹<https://www.cenelec.eu/dyn/www/f?p=104:110:343628176220501>

Layers

Network protocols respectively network stacks are build up in different abstract layers. This abstraction provides a clean separation of businesses and hides the implementation details of other layers. It also drastically ameliorates flexibility, because it allows the replacement of implementations on the layers level. One can easily replace one layer, e.g. replacing Ethernet with 802.11, without affecting those on top. The ISO/OSI reference model describes the different layers. It separates networks into seven different layers, visible in Figure 2.2. [Col17]

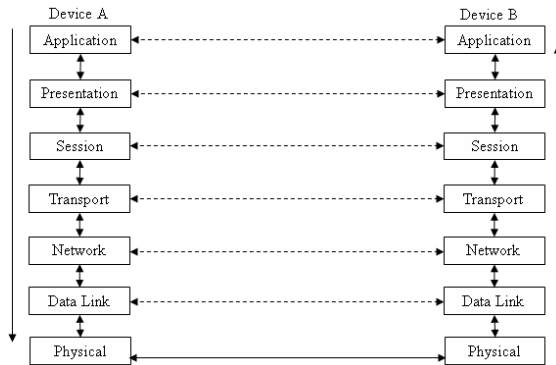


Figure 2.2: The ISO/OSI reference model
 Source: https://www.researchgate.net/figure/ISO-OSI-reference-model_fig5_325153525, accessed on 27. Jan 2019 19:25

This model can also be applied to KNX, as visible in Figure 2.3. KNX implements layer one, two, three, four and seven. Layer one and two will be discussed in the section 2.2

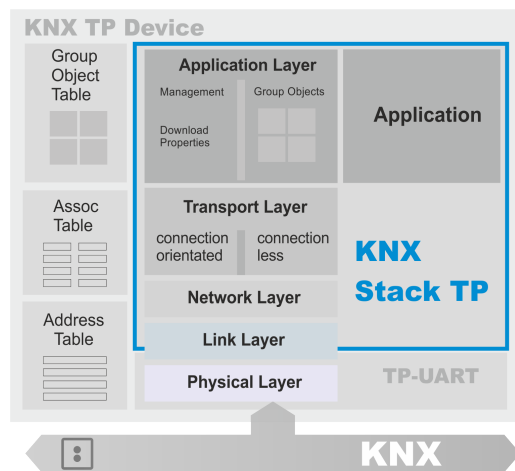


Figure 2.3: OSI Layers of KNX
 Source: <https://www.weinzierl.de/index.php/de/alles-knx1/knx-stacks/knx-stack-overview>, accessed on 27. Jan 2019 19:29

“Transmission media”. The section 2.2.5 “Frames” will discuss layer three and four. The highest layer, i.e. seven, will be discussed in section 2.2.7 “Interworking”. [Hüb18][MW16][DIN04d]

Transmission media

KNX offers different transmission media. KNX/TP, KNX over twisted pair, transmits both the data and a low power energy via one pair of drilled wires. It allows different wirings including stars, line and trees, which may also be used in combination with one another. There exist two different versions of KNX/TP. KNX/TP0, which was directly inherited from Batibus, and KNX/TP1, which was introduced by EIB. KNX/TP0 is considered outdated and expected to slowly roll out, while KNX/TP1 is the current standard. KNX/TP1 has the biggest market share, being supported by over 90% of KNX products. It allows for high quality transmissions at a low hardware cost. KNX/TP1 will be further explored in the following subsection 2.2.1 “KNX/TP1”. Another transmission medium is KNX/PL, KNX over Powerline, permitting the use of existing low voltage networks, 230V respectively 400V, to transmit the KNX protocol. It accommodates a bandwidth of up to 1,2kBit/s. KNX/RF, KNX over radio, providing communication via radio transmission, which can be handy, when installing additional wires is not an option. Last but not least is KNX/IP, KNX over Internet Protocol. It uses standard ethernet technology and allows for standard transmission bandwidth of 10Mbit/s respectively 100Mbit/s, subject to the Ethernet technology used on the subsequent layer. KNX/TP1 is the most frequently used transmission medium. Due to its popularity and the lack of availability of other devices at hand, this master thesis will only discuss KNX/TP1 henceforth. While the derived implementation and findings are only to be considered valid for KNX/TP1, the described concept shall be applied to other transmission media as well. [MKW12][Hüb18][Sok17][KW06]

2.2.1 KNX/TP1

Segments

The smallest element in the structure of a KNX/TP1 network is the segment. It connects different KNX devices on a bus. KNX/TP1 distinguishes between KNX/TP1-64 and

KNX/TP1-256, only varying the number of devices that can be connected to one segment. A direct comparison is provided in Table 2.1. Nevertheless, one should always consider a segment to be KNX/TP1-64, owing to the fact that, as soon as there is a single KNX/TP1-64 device in a segment, the entire segment has to be treated as KNX/TP1-64. What is more aggravating is that, many vendors don't disclose whether they use KNX/TP1-64 or KNX/TP1-256. A segment can be wired in different ways. It is possible to use a linear, star, tree topology or a mix of them all. The only prohibited topology are circles. Various topologies are visualised in Figure 2.4. [Sok17][DINo4d]

Characteristics	Description TP1-64	Description TP1-256
Medium	Shielded twisted pair	
Topology	Linear, star, tree or mixed	
Baud rate	9 600 bps	
Device supplying	Normal: bus powered devices - optional: remote powered devices	
Device power consumption	3 mA to 12 mA	
Power Supply Unit (PSU)	DC 30 V	
Number of PSUs per physical segment	Maximum 2	
Number of connectable devices per physical segment	Maximum 64	Maximum 256
Number of addressable devices per physical segment	Maximum 256	Maximum 256
Total cable length per physical segment	Maximum 1 000 m	
Distance between two devices	Maximum 700 m	
Total number of devices in a network	More than 65 000 (with bridges)	More than 65 000

Table 2.1: Comparison of KNX/TP1-64 and KNX/TP1-256
Source: DIN EN 50090-5-2 page 37

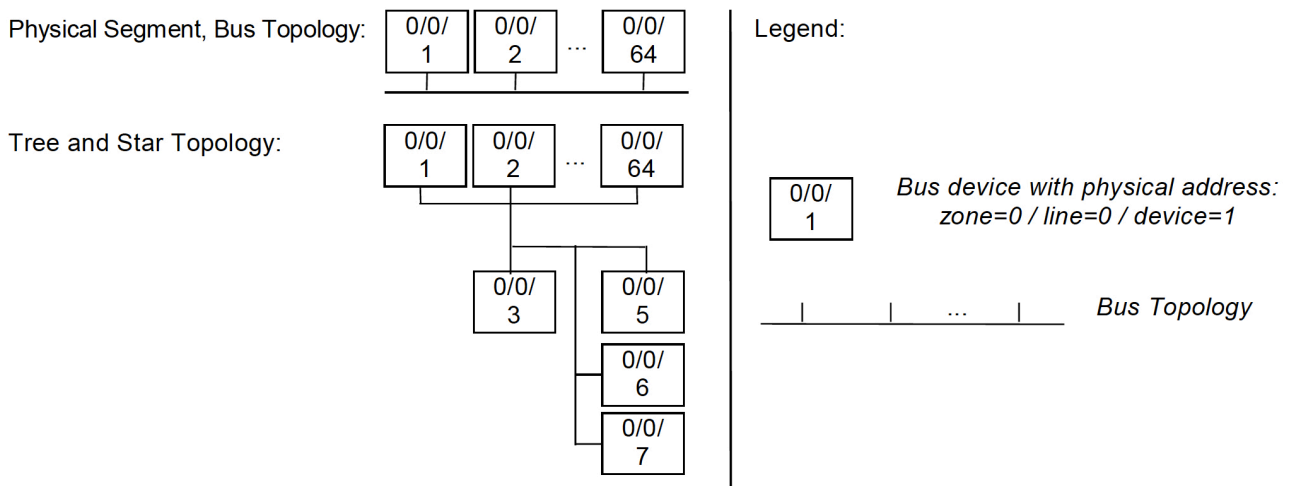


Figure 2.4: Physical segments of a KNX/TP1 network in different arrangements
Source: DIN EN 50090-5-2 page 53

Bus signal

One advantage of KNX is that only loads need a dedicated power supply. The field bus provides both electricity to bus devices and transmits data at the same time. The dedicated power supply and the field bus are galvanically isolated. The KNX field bus comprises of two wires. A positive red wire, sometimes called an A line, and a negative black wire, sometimes called a B line. The data is converted to a voltage signal. Each segment needs at least one PSU, providing a Direct current (DC) at a nominal voltage of 24V. This decentralisation of PSUs disallows local failures from affecting the entire network. Line couplers and bridges being the very same component, just with a different configuration, are always powered by the PSU on the subordinate line. Line couplers and bridges

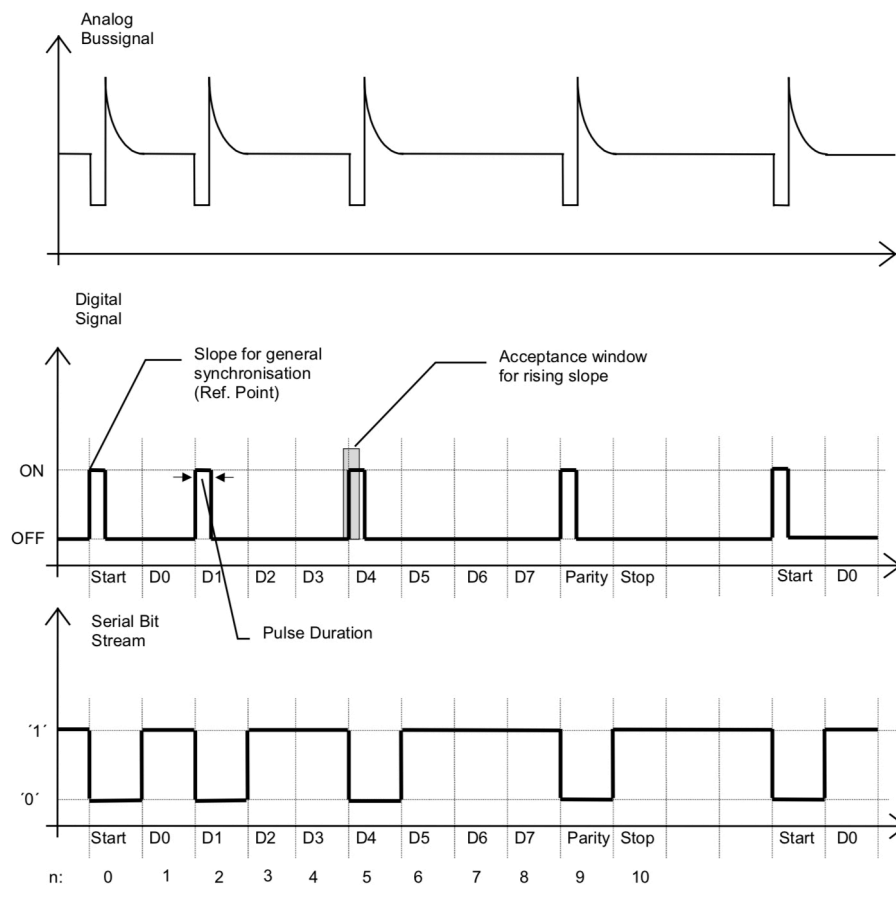


Figure 2.5: Relation of digital signal and serial bit stream

Source: DIN EN 50090-5-2 page 49

also function as an isolation between different segments. They will only transmit the data, but not the DC-part. [Kas+05][Hüb18][Sok17][Ass13]

A logical zero corresponds to a voltage drop of the bus signal for about $35\mu\text{s}$, which is followed by an equalisation of the voltage, during which the voltage is higher than the normal DC-part. The higher voltage provides the energy, that was consumed during the activity. In contrast, a logical one is encoded as the idle state. The signal will be equal to the DC-part provided by the PSU. Consequentially, a logical one is indistinguishable to bus silence. To allow receivers to recognise a transmission starting with a logical one, serial characters, as explained later in this section, always start with a start-bit of a logical value zero. [Hüb18] [DIN04d]

The analogue signal of a serial character is illustrated in Figure 2.5.

Bus access

It is possible to distinguish bus access between deterministic and stochastic. Two deterministic bus access schemes include Master-Slave and Token-passing. Within a Master-Slave scheme, there is one master node that gives permission to use the bus to other nodes, called slaves. When using a Token-passing scheme, a token, giving permission to use the bus, is passed among nodes. Other bus access schemes, that are considered stochastic, allow multiple access at a time and try to detect or avoid collision. They are called Carrier Sense Multiple Access / Collision Detection (CSMA/CD) and Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) respectively. KNX/TP1 uses CSMA/CA. As a device starts sending its data, it also listens to the signals being transmitted on the bus. As soon as the device notices a difference between the signal being sent by itself and the signal it receives on the bus, it will immediately pause its transmission. The other device in contrast will keep sending its data, because it never detected a mismatch on the bus. Due to the encoding of data in the analogue signal, a logical zero will always supersede a logical one. The next bus access will happen after an arbitrary waiting period, in order to prevent repeated collisions. [Sok17]

Another means to control bus access are priority classes, explained further in subsection 2.2.5 “Frames”. Frames with an access class of one, are sent immediately after a bus silence of 5,2 milliseconds. Class two frames are sent after another rest period of 312 microseconds.

Serial Characters

KNX/TP1 does not transmit frames as one big block, but inside small units called Serial characters. Serial characters transmit 8 data bits at a time, but are comprised of more bits to control transmission. The first bit of a serial character is called a start bit. It is always set to zero, meaning it sends a bus signal. The start bit is followed by the data bits, starting with the Least Significant Bit (LSB). Each serial character terminates with a parity bit and a stop bit, which will always be set to one. There will be at least two pause bits between each serial character, resulting in each of them taking at least 13 bit times to send. The explained structure is illustrated in Figure 2.6.[DINo4d] [Hüb18] [Sok17]

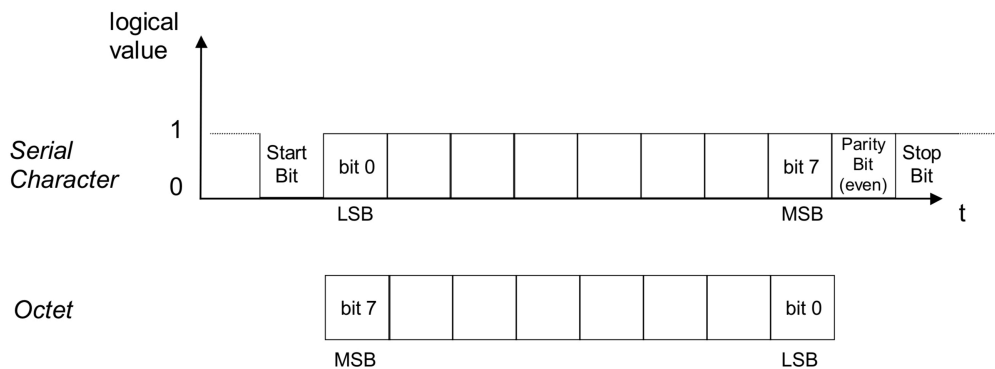


Figure 2.6: An octet mapped to a serial character
 Source: DIN EN 50090-5-2 page 38

2.2.2 Topology

This subsection describes the logical topology of a KNX network. It is closely related to the physical topology and the segments explained in subsection 2.2.1 “KNX/TP1”, but provides greater abstraction. KNX follows a hierarchical structure. The smallest element being a segment, building up to a line, an area and a network.

Line

A line constitutes one to four segments, which are interconnected by a bridge. It can address up to 256 devices. The KNX devices itself are addressed by the numbers 1 to 255. The coupler, connecting the line to an area, is addressed by 0. Bridges, that have to be used when addressing more than 64 devices in a KNX/TP1-64 segment, require their own address and limit the number of available addresses for other KNX devices. While technically allowed to put 3 bridges in a row, it is not recommend, since there may be at most six couplers between two devices that exchange frames. In order to mitigate this problem, it is possible to use a tree structure, as shown in Figure 2.4.[Kas+05][Hüb18][Sok17][Ass13]

Zone (Area)

The higher level is called a zone, sometimes also referred to as an area. One superordinate line, called the main line, connects up to fifteen different subordinate lines. The main line has the line address of zero. All the others are numbered 1 through 15. Strictly speaking, it is possible to connect ordinary KNX devices other than line couplers to the main line. Although, it is not recommended and considered bad practice. The line address zero combined with the device address zero is reserved for the backbone-coupler, connecting this area to others.[Kas+05][Hüb18][Sok17][Ass13]

Network

The highest level of configuration in a KNX network, is that of connecting 16 different zones to a network. The line connecting all zones is called a backbone line. Nonetheless, inspite of it being a possibility to connect ordinary KNX devices to the backbone line, akin to common practices within zones, it is considered unwise. [Kas+05][Hüb18][Sok17][Ass13]

A compact overview of the maximum number of devices and maximum distances for lines, zones and networks can be found in Table 2.2

2.2.3 Addressing

KNX devices can be assigned two different kinds of addresses - one physical address and multiple logical group addresses. In order to be able to differentiate one from the other, the parts of a physical address are separated by a dot, while those of a group address are separated by a slash. Figure 2.8 shows an example of a topology with physical and logical addresses assigned.

Physical / Individual

Every KNX device has to be assigned a unique physical address. It has to have a subordinate ID of the line. During commissioning, this physical address will be written to the device's Electrically Erasable Programmable Read-Only-Memory (EEPROM). A physical address contains three parts: 4 bits to encode the area, 4 bits to encode the line and 8 bits to encode the device address. A visualisation can be found in Figure 2.3. Certain addresses are reserved for special cases: Addresses with a device address set to zero must be assigned to line couplers. Addresses with both line address and device address set to zero must be assigned to backbone-couplers. The address 0.0.0 is reserved for later use. Physical addresses are mostly used for management purposes and do not hold any significance for ordinary traffic. [Sok17] [DIN04c] [Hüb18]

Logical / Group

There are two conventions to encode group addresses: two-level addressing and three-level addressing. However, group addresses do not have to follow this convention and can be freely configured by the administrator. The two-level addressing consists of five bits for the main group, allowing up to 32 main groups, and eleven bits for subgroups, allowing up to 2048 subgroups per main group. The three-level addressing starts with five bits for the main group, followed by three bits for the middle group and eight bits for the subgroup. The group address 0, respectively 0/0 and 0/0/0 are used for broadcast communication. The structure of the different addressing schemes is displayed in table 2.4 and table 2.5. [Sok17] [DIN04c]

CHAPTER 2. BACKGROUND AND PRIOR WORK

Individual address															
Octet 0								Octet 1							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Area address				Line address				Device address							
Subnetwork address															

Table 2.3: Structure of physical address

Source: DIN 50090-4-2 page 5

Group address															
Octet 0								Octet 1							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Main Group				Sub Group											

Table 2.4: Structure of group addresses with 2 levels

Source: KNX für die Gebäudesystemtechnik in Wohn- und Zweckbau, page 71

Group address															
Octet 0								Octet 1							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Main Group				Middle Group				Sub Group							

Table 2.5: Structure of group addresses with 3 levels

Source: KNX für die Gebäudesystemtechnik in Wohn- und Zweckbau, page 71

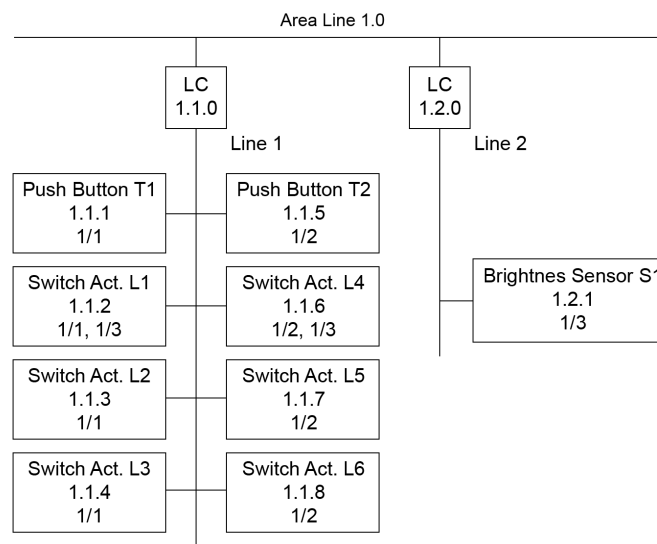


Figure 2.8: Example topology with physical and logical addresses

Source: KNX für die Gebäudesystemtechnik in Wohn- und Zweckbau, page 73 (translated)

2.2.4 Communication objects

DIN ISO 50090 3.2 specifies two different object models: Group objects (also called communication objects) and interface objects. The latter is exclusively used for special services like management and not in group communication. Conceptually, communication objects can be described as an area inside a device’s memory. It is used for exchanging data between different devices and the applications running on them. Each communication object can be identified with multiple group-addresses, but may be at the most identified with only one sending group-address. Communication objects have to obey certain guidelines, visualised in Figure 2.9. The object type describes the data being transported inside the communication object and ranges from 1 bit to 14 bytes. Only objects with the same object type can be linked together. In order to provide interoperability, DIN 50090 3-3 specifies various datapoint types, that have to be used in order for the vendor to be able to certify his devices. Datapoint types will be further discussed in the subsection Interworking. The priority may only be set to “urgent”, “normal” and “low”. “System” is not allowed for group-object communication. Configuration flags are used to configure access to the group object of a device. Different configuration flags are explained in Table 2.6. Communication flags represent the internal state of a group object and are used to communicate between the user application and the group object server, also referred to as user process. [Sok17][Hüb18][DIN04a][DIN09]

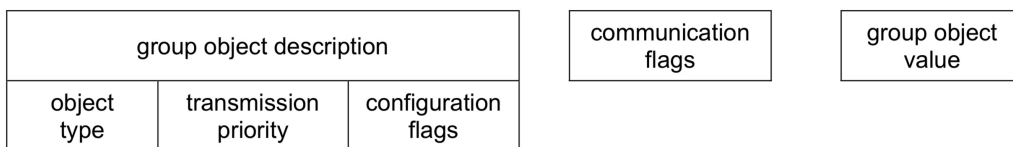


Figure 2.9: Structure of a group object

Source: DIN 50090-3-2 page 7

configuration flag	If enabled ...
read enable	respond to read requests on the field-bus
write enable	update group value as result of incoming frame on field-bus
transmit enable	write group value to the field-bus if changed internally
update enable	responses to poll-frames are treated as write requests
communication enable	acknowledge frames and change internal value, disabling also disables all other flags

Table 2.6: Meaning of configuration flags

Source: DIN 50090-3-2 page 8

2.2.5 Frames

Frames are the central component of every communication in KNX. There are three different frame types, serving all communication modes: data frames, poll frames and acknowledgement frames.

Data frames

There are two different kinds of data frames: standard and extended. Standard frames have a length of 9 to 23 bytes, including up to 16 bytes of user data. A standard frame is portrayed in Figure 2.10. Including the acknowledgement, transmission of a standard frame on a KNX/TP1 network will take between 20 and 40 milliseconds. [DIN04d]

Standard frames are structured as follows: The first byte is the control field. It contains the frame type, a repeat flag and the frame's priority. The frame-type of a standard frame is set to 1. The second and third octet of the standard frame contains the physical address of the sender. The physical address is followed by the two octets to encode the destination address, either being a logical or a physical one. The type of the destination address is encoded in first bit of the sixth octet. One represents a group address, zero a physical address. The sixth octet also contains the hop count and the length of the payload. The seventh contains the Transport layer Protocol Control Information (TPCI), which will always be set to zero on a KNX/TP1 network. The TPCI is followed by the Application layer Protocol Control Information (APCI) and the communication object respectively an interface object. APCI is used to encode different services. Ordinary group messages rely on three services: "A_GroupValue_Read", "A_GroupValue_Response", "A_GroupValue_Write", visualised in Table 2.7. The final octet is the check octet, containing parity bits to ensure data correctness. [DIN04d] [Sok17]

Extended frames can transport up to 255 bytes of user data. The structure of an extended frame is shown in 2.11. In contrast to standard frames, extended frames have two control fields. The first one is identical to that of a standard frame, except that the frame type is set to zero. The second control field contains the address type of the destination address, the hop count and 4 bits for the Extended Frame Format (EFF), which is mostly used in the context of HVAC. Similar to the standard frame, the control fields are followed by source and destination addresses. The seventh octet provides an entire 8 bit to encode the length of the payload. The remaining are identical to that of a standard frame. [Sok17]

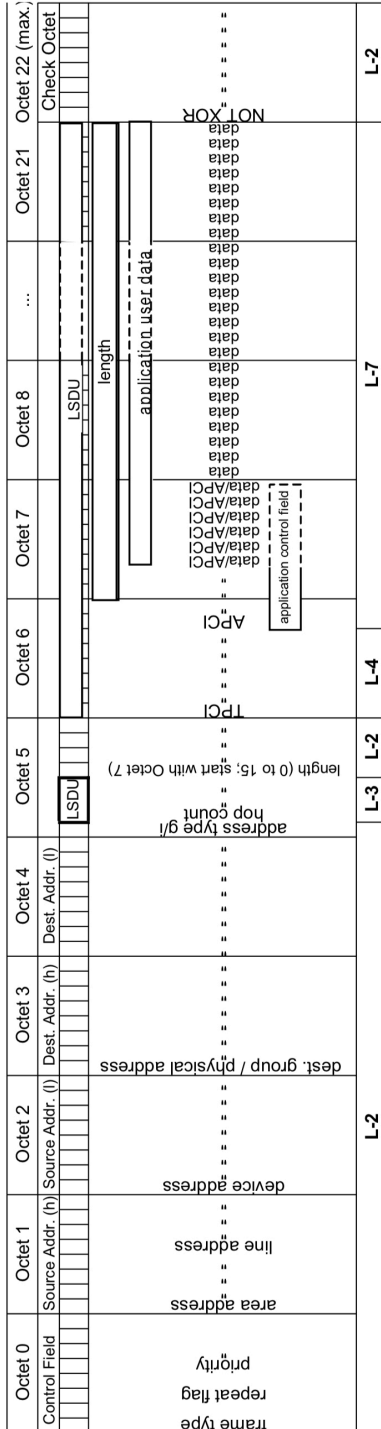


Figure 2.10: Structure of standard data frame

Source: DIN 50090-5-2, page 60

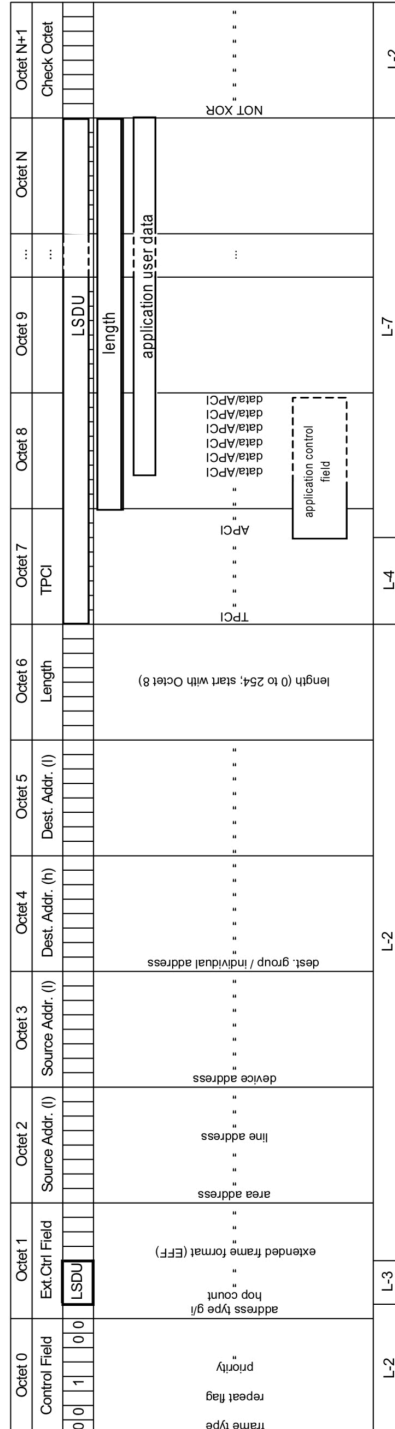


Figure 2.11: Structure of extended data frame

Source: DIN 50090-5-2, page 61

APCI														Application layer Service			
Octet 7							Octet 8										
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1		
TPCI							APCI				data						
						0	0	0	0								A_GroupValue_Read
						0	0	0	1								A_GroupValue_Response
						0	0	1	0								A_GroupValue_Write

Table 2.7: Group object related application layer services
 Source: DIN 50090-4-1, page 9

Poll frames

The structure of the initial poll frame is akin to that of a standard data frame. One master node requests data from up to 15 different slaves, which are responding with exactly one serial character each, in their corresponding time-slot. Each slave is assigned the group address they essentially answer to and their time slot to answer during configuration time. When using the KNX/TP1 transmission medium, poll frames will only be transmitted inside a single physical segment and are not forwarded to other segments. Figure 2.12 presents the structure of a poll-frame and figure 2.13 the response scheme. [DIN04d].

Acknowledgement

Acknowledgement frames consist of one single control field (see Figure 2.14). They can only be sent in response to data frames. Acknowledgement frames can encode four different states: “ACK” for acknowledgement, “NACK” for negative acknowledgement, indicating a broken frame. “BUSY” represents a successful transmission of the frame, but the receiver is requesting to resend the frame at a later point in time, due to the fact that it is busy with another task at the moment. “NACK“ and “BUSY“ can also be combined together into “NACK+BUSY”. The speciality of acknowledgement frames is that all receivers answer at the same time. Due to the nature of the underlying KNX/TP1 network, the zero of a “NACK” in b7 and b6, will always supersede the ones of “ACK”. The same applies to the zeros of “BUSY“ in b3 and b2. In case the receiver is on a different line, the coupler will forward the frame and acknowledge it locally. [Sok17]

Octet 0								
Short ACK								
b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	1	0	0	1	1	0	0	ACK
0	0	0	0	1	1	0	0	NACK
1	1	0	0	0	0	0	0	BUSY
0	0	0	0	0	0	0	0	NACK + BUSY

Figure 2.14: Structure of acknowledgement frame
 Source: DIN 50090-5-2, page 64

Priority and Repeating

KNX defines two different access classes. Access class 1, representing important traffic, contains the priority “system” and “urgent”. Access class 2 contains the other two priorities “normal” and “low”. These priorities are influenced by the priority configured for a communication object. Access class 1 frames are sent directly after 50 bits of bus silence, while access class 2 frames have to wait for another 3 bit times. Class 1 frames will always take precedence. A class 2 frame, that needs to be retransmitted due to a negative acknowledgement, will automatically turn into an access class 1 frame. [Hüb18]

Routing counter

The routing counter, also called hop count, is encoded in 3 bits, which allows for 8 different values: zero to seven. If the hop count is set to zero, couplers won't forward the frame, rather, simply drop it. A frame with any hop count between 1 and 6 will be forwarded and the hop count will be decremented by 1. In case of the hop count being set to 7, the frame will be infinitely forwarded, without decrementing the hop count. This signifies that, depending on the couplers filter rules, a frame will be forwarded into every line of a KNX network. It is only meant to be used for management and cannot be set for ordinary devices in ETS. [Sok17] [DIN04c]

Parity byte

In addition to every character having its own parity bit, each frame contains a check octet to ensure that the transmission was impeccable. The parity bit of a character is chosen in a manner that the sum of all one bits is even. The check octet of a frame contains a parity bit for every first, every second, ..., every eighth bit of each octet. The parity bit is chosen in a manner that the number of all one bits is uneven. [Sok17]

2.2.6 Communication

KNX provides different communication modes, that also differ in the services that they define. The various modes are: point-to-multipoint (connection-less), point-to-domain (connection-less), point-to-all-points (connection-less) and point-to-point (connection-less and connection-oriented). [DIN04b]

Visualisation

Visualisation in KNX can be described as the act of eavesdropping. Visualisation devices must to be positioned at a central position, often times main lines and backbone lines, and hear in on all traffic on the bus to register changes to group-addresses they are concerned with. This requires frames to be forwarded to the segment containing the visualisation device, potentially resulting in more traffic on main lines as well as backbone lines. For certain applications, e.g. dimming, it is insufficient to just eavesdrop on the bus. Instead, the visualisation device will have to query data from the corresponding device. One important pitfall in this context is that devices will always respond with their sending group address for a communication object, even if the data was requested on a different group address. [Sok17]

Management

DIN 50090 7-1 determines certain network and device management procedures. They permit the reading and alteration of device and application parameters and provide special services, including but not limited to restarting and resetting devices. [DIN04e]

2.2.7 Interworking

DIN 50090 3-3 defines the KNX interworking model, that enables devices from different vendors to work together as one ensemble. It follows a definite hierarchical structure, seen in Figure 2.16.

In context of the KNX interworking model, the term application does not refer to a program running on a single device, but on the contrary to the sum of functional blocks divided among various devices, interconnected by the bus system, serving one common purpose to solve a task as a whole. The next level, Functional blocks, are defined based on two elements: a set of datapoint types as well as a well-defined functionality. Each functional block defines its I/O datapoints, its physical I/O and how they interact. [Hüb18][DIN09]

The central components of KNX Interworking are standardised datapoint types, allowing different devices to interact by using identical encoding for data. Annex A of DIN 50090 3-3 specifies various datapoint types.

Two example datapoint type definitions can be seen in Figure 3.3 and Figure 2.19.

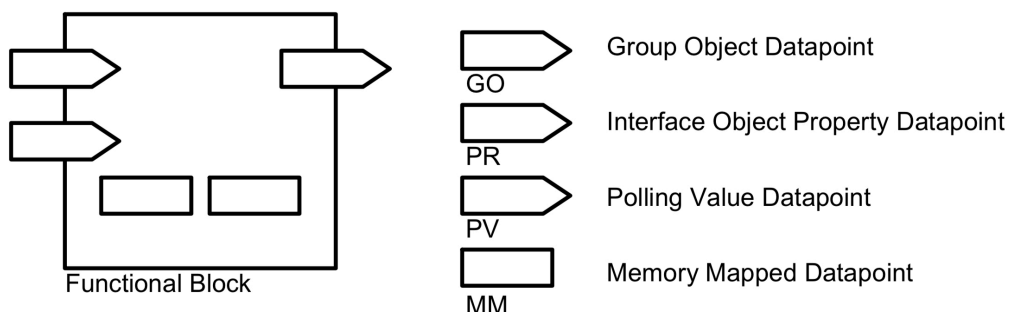


Figure 2.15: Visualisation of functional blocks
 Source: DIN 50090-3-3, page 10

CHAPTER 2. BACKGROUND AND PRIOR WORK

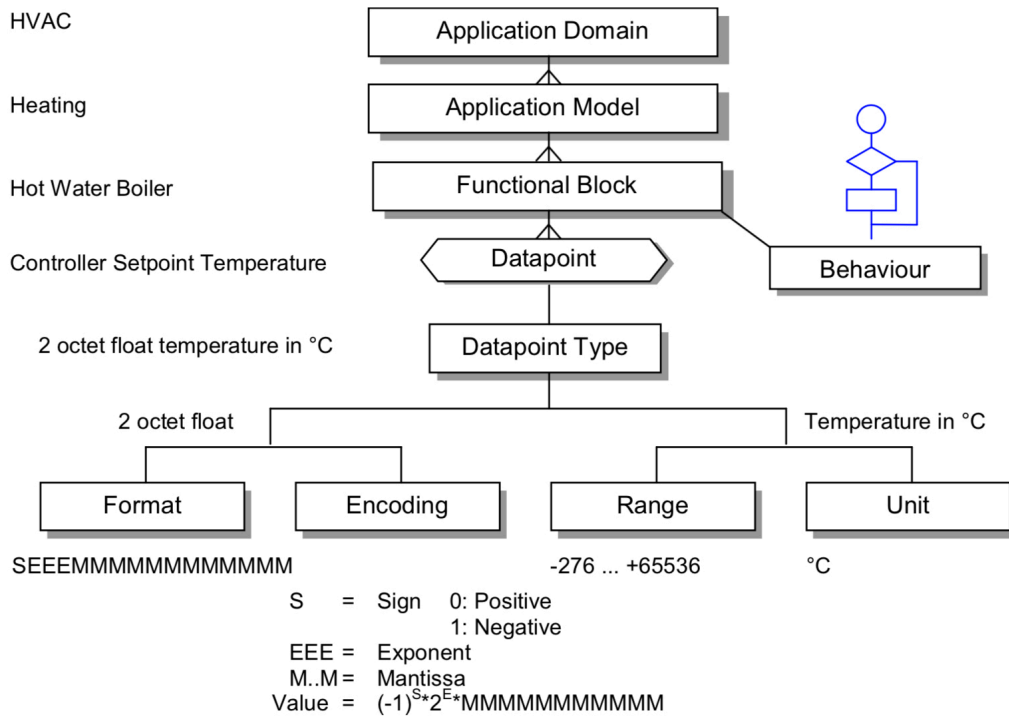


Figure 2.16: Example of the KNX interworking model

Source: DIN 50090-3-3, page 11

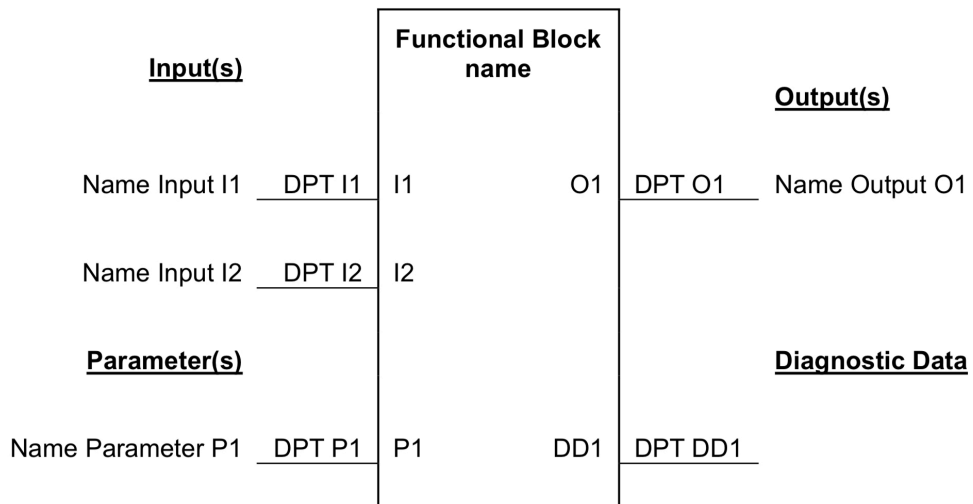


Figure 2.17: Standard representation of functional blocks

Source: DIN 50090-3-3, page 8

2.2.8 ETS

ETS, maintained by EIBA is the only tool available to configure KNX installations, that's virtually used by all electricians managing them. It is used to manage a virtual representation of the building, which allows to configure devices including their topology in the system, assign them different group addresses and set up their functionality. It also provides reporting and diagnostic features.[MKW12][Kas+05][Sok17]

2.2.9 Official Extensions

The KNX association published two extensions that are concerned with its security: KNX Data Secure and KNX IP Secure.

KNX Data Secure assures integrity, freshness and confidentiality. It can be integrated into existing installations, but requires all participants of a group communication to support KNX Data Secure. All devices are shipped with a Factory Device Set up Key (FDSK). The technician who sets up the KNX installation, has to enter this FDSK into ETS, which will generate device specific keys and load them onto the device. Moreover, ETS will also generate shared group keys for group communication.

KNX IP Secure provides an extra layer of security when using KNX/IP as transmission medium. [Ass] [Sok17]

2.3 Network Filters

2.3.1 Terminology

Network Security Through Data Analysis: From Data to Action introduces a terminology to describe, what a sensor can see and how it deals with certain events. The most important term mentioned is "Vantage". Vantage describes the position of a sensor within a

network. Sensors with a different vantage will be able to observe different traffic. The second crucial terminology is “Domain”. Domain is the kind of information provided by a sensor. *Network Security Through Data Analysis: From Data to Action* distinguishes between three different domains: “Host” characterises events that can be observed on the host, including but not limited to logins, logouts or file access. “Service” refers to the events that can be observed by a service like the *access log* of an apache webserver. “Network” represents all the traffic that can be monitored on a network. The book gives an example of why different domains may be important: “The host sensor can record the last time a file was accessed, but does not relate that file to a URL or request. The service sensor can say that an HTTP session took place and include what page was served, but it will not record a half-open scan on port 80.” [Nor05, p. 8] The third and last term introduced is “Action”. Actions can be “report”, reporting all traffic to a central entity, “event”, reporting only selected events to a central entity, or “control”, to modify and block the traffic. [Col17]

2.3.2 Packet Filtering

Packet filtering is a general term to describe all means of controlling access to a network. It is used to decide whether a packet, coming from within or outside the network, is allowed to cross the filter. The most customary position of a filter is inside a router, ordinarily the router that connects an internal network to the external internet. [Nor05]

One can distinguish between ingress and egress filtering. Ingress signifying packages coming into a network, egress are those leaving a network. E.g. it is good sense to only let IP packets with a legitimate source address leave ones network. [Nor05]

There exist different approaches to categorise packet filters. One aspect is the maximum ISO/OSI layer inspected. Simple packet filtering will not go deeper than layer 3, inspecting the minimum header fields necessary for routing. So called shallow packet filtering goes a layer deeper, inspecting the transport protocol in use, including its port number. Deep Packet Inspection inspects the entire packet, all the way down to layer 7 in the ISO/OSI model. [Nor05][Bed10]

Another aspect of packet filtering is whether or not the filter keeps track of existing connections and its state. This is most often realised as a table, every row containing information about one session. "A state table entry is created when a connection is started out through the stateful device. Then, when traffic returns, the device compares the packet's information to the state table information to determine whether it is part of a currently logged communication session." [Nor05, Chapter 3.2]

Certain problems are faced on occasion in context to IP networks : The ability to define custom routes, called source routing, becomes an issue when used to hide a spoofing attack. It is not uncommon for devices to have multiple network interfaces that are able to communicate via IP. Hence it is not guaranteed, that traffic coming into a device will be routed much like traffic leaving a client, requiring shared states for effective clustered firewalls. Communication is a two-way street. Simple non-state-capable packet filters will even struggle with simple stateful communication. Either they block too much incoming traffic or leave behind a wide surface for attack with open ports. [Nor05] [Col17]

Implementations

Popular implementations of packet filters are IPTables² and Cisco Access Control list (ACL)³. The configuration of IPTables and Cisco ACL can basically be described as a list of rules. These rules consist of a policy, a source-address, a destination address, a protocol and additional options. The policy defines whether to forward ("permit" in Cisco ACL, "ACCEPT" in IPTables) or whether to block a packet ("deny" in Cisco ACL, "DROP" in IPTables). Both IPTables and Cisco ACL provide state-full capabilities with the "related" respectively the "ESTABLISHED" keyword. [Nor05]

An important aspect of creating rules is the order. Rules are evaluated starting from the top to the bottom, requiring specific rules to come in at the top and more general rules to come in at the bottom. [Nor05]

²<https://netfilter.org>

³<https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html>

2.4 Deep Packet Inspection

Deep Packet Inspection is concerned with the content of a packet. It allows to monitor, prioritise or even manipulate data transmission in real time. While inspections on lower ISO/OSI layers allow the inspection of metadata, e.g. who is talking to whom, Deep Packet Inspection allows the examination of the actual content. This content is commonly matched against a set of signatures to detect attacks, e.g. certain parameters that are known to cause a crash, or simply the signature of a virus / worm. In real world scenarios, it is also often used to prevent access to content illegal in a certain jurisdiction or to allow lawful interception. [Nor05][Bed10]

In Germany, Deep Packet Inspection conflicts with the right to informational self-determination, the merits of use have to be carefully considered. [Bed10]

There are three common approaches on implementing Deep Packet Inspection: String Searching Algorithms to find exact matches. A more elaborate approach is regular expressions, providing more flexibility by allowing wildcards, length restrictions and various patterns. Last but not least, Finite state machines, provide a promising approach to match a stream of data against a set of signatures. While regular expressions are the most common format to define rules for Deep Packet Inspection, they can be converted to finite state machines. Finite state machines are differentiated into Deterministic Finite Automata (DFA) and Non deterministic Finite Automata (NFA), both having benefits and drawbacks. DFAs suffer from a state explosion, requiring too much memory, while NFAs cannot provide a limit on simultaneously active states. Jignesh D. Patel noted in his master thesis “Algorithms for deep packet inspection”: “While the DFA for any one RE is typically small, the DFA that corresponds to the entire set of REs is usually too large to be constructed or deployed.” [CS11, Abstract]

Sailesh Kumar et. al proposed an enhancement to DFA in their publication “Algorithms for deep packet inspection” called D^2FA , Delayed Input DFA. They observed: “groups of states in a DFA often have identical outgoing transitions and we can use this duplicate information to reduce memory requirements.” [Kum+06, p. 340] By incrementally transforming a DFA, replacing and combining transitions into a default transition, they were able to reduce the number of transitions by more than 95%. [Kum+06]

In his master thesis, Jignesh D. Patel analysed different approaches to Deep Packet Inspection, postulating D^2FA to be promising and advanced it even further. His research goal was to provide an implementation of regular expressions using DFA that are supposed to achieve high throughput and a low memory consumption. Their experiments indicated that their new approach OD^2FA is extremely promising. [Pat12]

On another note, A. Munoz et al. developed the tool “Snort2Regex”, converting rules of the intrusion prevention system Snort, similar to filter rules explained in the previous 2.3.2 “Packet Filtering” section, into simple regular expressions, that can be used outside the context of Snort. They propose a “meta-rule-format” in the form of regular expressions, filling simple regular expression templates with the corresponding values from Snort rules. [Mun+11]

2.5 Prior work

2.5.1 Unofficial KNX extensions

Thomas Novak et al proposed a lifecycle model in their publication “Safety- and Security-Critical Services in Building Automation and Control Systems.” Their paper aims at providing a security concept in its entirety, from the first step of planning until decommissioning. By equipping selected nodes with smart cards, it enables them to assure authenticity, integrity and confidentiality for selected communication. Within their proposal, nodes with smart card capabilities communicate over the same bus as nodes without smart card capabilities, permitting them to fit devices with the capacity to communicate securely only where needed. Different phases include the development phase, consisting of concept, assessment of safety-security requirements and realisation, and the use phase. The latter incorporates installation, an evaluation of the previously made safety-security requirements, the concrete operation and necessary maintenance. The use phase is concluded by decommissioning the system.

Based on the prior work “Secure EIB”, developed at the TU München, Wolfgang Granzer et al proposed a new KNX extension called “EIBSec”, described in their publication “Security in networked building automation systems.” With respect to the limited computing power of embedded devices, Wolfgang Granzer et al decided against asymmetric

cryptography and preferred Advanced Encryption Standard (AES). Using so called advanced coupler units, the presented concept proposes a decentralised scheme for distributing keys. Different communication modes require different keys, namely session-keys for point-to-point communication and group-keys for point-to-multicast communication. Secure communication between nodes and their concerned advanced coupler units is ensured by assigning each non-coupler device with a node key. Counter mode provides measurement against replay attacks.

2.5.2 KNX-related research

The chairperson of the Chair for Information and Communication Services at the University of Rostock hosted many research projects in connection to KNX. One such student's work that shall be highlighted with respect to the later use of his findings in section 5.3 "Thought experiment" is the Master thesis *Sicherheitsanalyse von Gebäudeautomationsnetzen auf Feldebene am Beispiel von KNX* by Johannes Goltz.

Johannes Goltz analyses different attacks on BASs and provides a classification for the risk of single devices or even entire lines. The Goltz-factor is divided into two subfactors, namely - the Static Goltz Factor and the Dynamic Goltz Factor. Different facets like risk-potential, accessibility of the device, accessibility of the bus and access to other devices on the bus are summarised in the Static Goltz-factor. The Dynamic Goltz-factor deals with the amount and kind of traffic, analysing the bus-load and distinguishing between simple group and management communication. [Gol18]

Concept

Contents

3.1	Extended Coupler Unit	36
3.2	Metadata on the bus	39
3.3	Data in ETS	40
3.4	Filter language	42
3.5	Matching	43
3.6	State	47
3.7	Generating rules	48
3.8	Building matchers for DPTs	54
3.9	Example	55
3.10	Attacks	58
3.11	Rule parsing	59

This chapter will describe the proposed concept of a Deep Packet Inspection system for field busses. In the context of field bus systems, DPI systems will allow the inspection and control of the content being sent over the bus, from just limiting it to particular communication, e.g. a device may only invoke simple “Light off” / “Light on” calls, but no management procedures, up to limiting the actual content being sent, e.g. the motion

sensor may only turn on a light, but never turn it off, or the temperature sensor may only report values between -15°C to 30°C .

The first section 3.1 “Extended Coupler Unit” explains the extended coupler unit itself. In the subsequent section, a method for analysing the metadata on the bus will be presented and done exemplarily for KNX/TP1. The existing methods to extract data from ETS are examined and then an analysis shows what data can be extracted via the selected method. Together with the data on the bus, they build the basis for the proposed filtering language, defined in 3.4. Different aspects as matching and state are discussed in the succeeding sections. Section 3.7 introduces a novel algorithm to automatically build most-restrictive filters based on the data in ETS.

KNX Data Secure and KNX IP secure are considered out of scope due to their insignificance and minor marketshare.

3.1 Extended Coupler Unit

Couplers, whether they are line-couplers or backbone-couplers, constitute a central unit inside each KNX network. They figure prominently, because couplers are the unit that connect different lines and eventually build up entire networks. ETS itself is already able to generate basic filter rules, but they are limited to simple filtering based on group-addresses. By introducing the concept of an extended coupler unit, this master thesis proposes a replacement for coupler units that provide better filtering capabilities and a better means of dealing with both undesirable and even illegal traffic. At a fundamental level, it follows the concept of a standard coupler on KNX/TP1: “A router is a device that interconnects a hierarchically higher subnetwork and a hierarchically lower subnetwork.” [DINO4c, page 17] A visual illustration can be found in Figure 3.1. Following the terminology of *Network Security Through Data Analysis: From Data to Action*, the vantage of each coupler is the traffic on the subordinate and superordinate line. The domain of the perceived information is network. The action of the sensor is control. It will modify the traffic and optionally report policy violations. Extended coupler units can be used to replace either all couplers or just a selected subset to protect high-value targets, providing a compromise between cost and security.

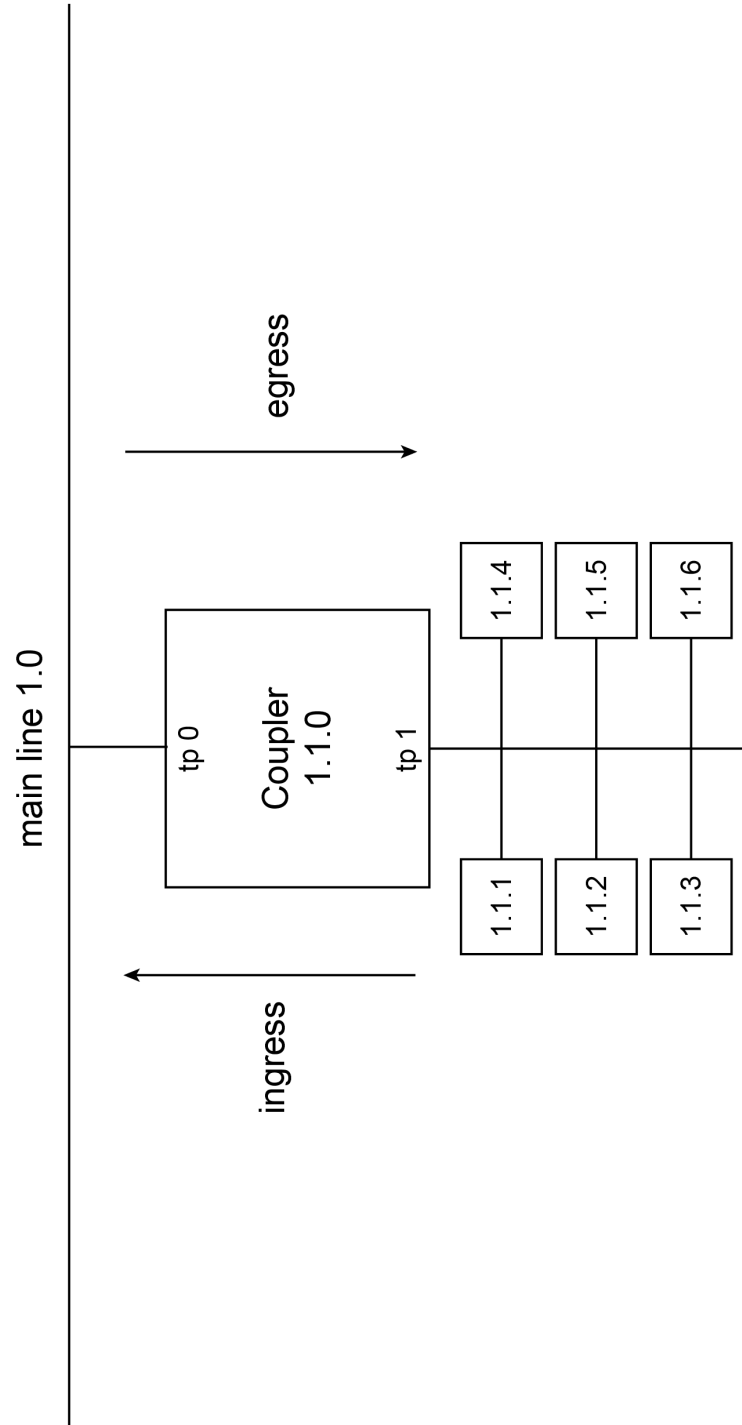


Figure 3.1: Simple illustration of a coupler

The KNX router is defined in DIN 50090 4-2 Sections 4.6.2 and 5.4.4. The behaviour of the Extended Coupler Unit is based on that of a standard KNX router with a few additions. The big difference appears on the network layer - layer 3. Unlike a standard KNX router, where only point-to-multipoint communications are run through a filter but all others are passed through, all traffic going through the Extended Coupler Unit has to pass through the filtering system. Much like a standard KNX router, the Extended Coupler Unit responds with Acknowledgment frames and does not forward Poll frames.

Section 5.4.4.2 “Detailed routing algorithm“ defines five different courses of action, a router shall respond with, seen in Table 3.1. Extended Coupler Units are fitted with one additional course of action, shown in italic. If a filter rule evaluates to DROP, the course of action shall be IGNORE_TOTALLY. If, on the contrary, a filter rule evaluates to NOISE, NOISE_LOCALLY shall be taken. By using noise, the router will attempt to interrupt an attackers frame. The transmission will be incomplete and bus devices will either completely ignore it or respond with a negative acknowledgement. There are different techniques to create noise on the bus. It is possible to flood the bus with logical zeros or send a random signal. The noise is sent for at least 1 octet. After, the router confirms that the device on the subordinate line stopped transmitting. In case it did not, the router sends noise for another octet and so on ad nauseam.

Extended Coupler Units are optionally fitted with an ethernet interface. Filter rules, generated by the tool introduced later in this chapter, will block all management traffic by default. This affects both management requests for devices on the other side of the network, but also management requests for the coupler itself. To provide management capabilities, the router will accept management requests to itself, but also to its subordinate line, via the ethernet interface.

Name	Description
ROUTE_UNMODIFIED	route from one subnetwork to the other without modification
ROUTE_DECREMENT	route from one subnetwork to the other and decrement the hop count value
FORWARD_LOCALLY	forward to local network stack of router
IGNORE_TOTALLY	ignore the frame, do not reply
IGNORE_ACKED	ignore the frame, but respond with an acknowledgement
<i>NOISE_LOCALLY</i>	<i>send noise to the subordinate line</i>

Table 3.1: Different courses of action for a frame

3.2 Metadata on the bus

In order to analyse the available metadata on the bus, it is necessary to inspect all different procedures for transport on all layers. KNX/TP1 provides two different procedures: Serial character and Frame. Besides the start bit and a parity bit for error correction, a serial character contains nothing but the partitioned frame from layers above. It does not provide any meaningful metadata. A frame comes in different variations: Data frames, standard and extended, poll frames and acknowledgement frames. Metadata available in frames is listed in table 3.2. For the sake of inclusivity it also lists poll-frames, but since they are not forwarded to other lines, they are of no relevance for the filters. Unlike in other ISO/OSI stacks like TCP/IP, where each layer adds its own addresses, e.g. MAC-Address, IP-Address, Port-number, KNX only knows physical and group addresses and uses that for addressing on all layers. One important piece of metadata that cannot be extracted from the frame itself is the datapoint type used. However, for a known datapoint point type, it is possible to verify that the data is plausible.

Layer	Frame types	Name
2	Data frame	Frame type
	Data frame	Repeat flag
	Data frame	Priority
	Data frame Poll frame	Src address
	Data frame	Dest address
	Data frame	Dest address type
	Data frame	Payload length
	Data frame Poll frame	Check octet
	Data frame (ext)	Extended frame format
	Poll frame	Poll group address
	Poll frame	number of expected poll data
	3	Data frame
4	Data frame	TPCI
7	Data frame	APCI
	Data frame	Data

Table 3.2: Metadata of frames

3.3 Data in ETS

It is difficult to formalise the extraction of available data from ETS, since the software is in development and subject to change. The latest available version as of mid February 2019 is ETS 5. The observations described hereafter were made on ETS 4 Professional. A copy of ETS 5 Professional was unavailable in the lab at the time. Different approaches exist to export data from ETS. It is possible for a human to manually extract all the data necessary from the ETS UI. Since this requires a lot of work, resulting in steep labour costs, and is likely to be very prone to error, the approach should be neglected. Furthermore ETS allows to export human readable report sheets, that contain all the information about the given installation. Similarly to the first approach, this is rather fallible and requires additional manual work. It would also be possible to develop a tool that scrapes the data from the pdf reports. ETS provides an export feature to two different machine-readable formats: `knxproj` and `OPC`. `knxproj`-Files are a zip-archive, that contain signatures and XML data, which are directly inherited from the database structure of ETS. An `OPC` export contains two files: An `EFS`-file listing all communication objects including their related group addresses and a `PHD` file, containing information about the general physical structure of the project. [Lou19] ETS itself stores its data in a Microsoft SQL Server (MsSQL) database. Directly accessing the MsSQL database has the advantage that the data is always up-to-date. There is no need to reexport the project after every modification. Moreover, due to the nature of a database, using MsSQL allows easier querying of specific device or group addresses in contrast to parsing the entire XML tree of a `knxproj` file.

For the purpose of generating rules it was necessary to have the data available in a machine-readable format. In consideration of the advantages of MsSQL outlined above, it was estimated to be the best approach to access the data. It is stored in a rather complex structure, in sixty-nine different tables. In order to gain an understanding of the database structure, it is very helpful to inspect the primary keys and foreign key relationships of each table. In addition, ETS creates stored procedures for common tasks. In most cases the SQL code of the stored procedures contains code comments and provides a great starting point for common use-cases.

An overview of the data extractable from ETS via its MsSQL database is given in table 3.3. Database tables that only refer to translated versions of text were omitted.

Name	Related tables	Description
Application	ApplicationBinaryData ApplicationProgramDynamic ApplicationProgramOption ApplicationSegment	Information about the application loaded onto the device
Physical address	Area Device Line	Structure of network and used devices
Building Structure	Building	Different aspects like rooms, parts of the building, etc.
Device Catalog	CatalogItem CatalogSection	Catalog of devices downloaded from vendor databases
Communication objects	CommunicationObject CommunicationObjectRef DeviceObject	Hierarchical structure of flags and datapoint type. DeviceObject is most specific, CommunicationObject most general
Status of loading program onto device	Trade	Empty, if loading program onto devices not in progress
Group addresses	GroupAddress	List of every single group address
Structure of group addresses	GroupRange	Definition of main and middle groups
Models	Hardware Manufacturer Product ProductAttribute	List of all different models of devices used in KNX network
Interface objects	InterfaceObjectProperty InterfaceObjectType	Interface objects used for point-to-point communication
Device configuration	various Parameter tables	Configuration parameters for each application
Relation of Communication objects and group addresses	Connector	Simple mapping from Group address to device objects and sending address
List of all Datapoint types	DatapointType DatapointSubtype	Used as a reference, not mapped to actual datapoint type field of tables
Status of exporting	various Export tables	Empty, if export not in progress
Projects	Installation Project ProjectHistory ProjectTrace	Different installations and projects managed by same ETS installation, history of previous modifications

41
Table 3.3: Data extractable from ETS

3.4 Filter language

Based on the metadata available on the bus and the data extracted from ETS, it is the aim to have rules that are as precise and unambiguous as possible, but yet compact and human-readable. Due to the fact that a human readable language was desired, using regular expressions to encode the entire rule was not an option. Nonetheless, a rule can be converted to a regular expression easily. The language design was inspired by the “POLICY PROTOCOL ADDRESS OPTIONS” pattern of Cisco ACL. Each rule requires at least four words: The policy, followed by the address of the sender. The third word must be the Destination type, followed by the destination address, which can either be a physical or logical address. The fourth word is optionally followed by other attributes, listed in Table 3.5. There are three different policies: ACCEPT to pass the filter successfully. This however, does not guarantee a transmission of the frame. If the hop count is set to 0, the frame will still be dropped by the router. The other policies are DROP and NOISE. DROP will result in the frame to be ignored completely, while NOISE will disturb the transmission. While it is possible to simply not forward forbidden frames, it is still desirable to prevent them even on the subordinate line of a router. This is where NOISE is used. Interrupting frames during their transmission requires them to be matched against the rules while they are still coming in. This can be implemented by converting the rule list to an automata, as described in the section 2.5 “Prior work”. The large majority of attributes are optional. In case an attribute is not set, it will act as a wildcard. E.g. a FrameType that’s not set will match both standard and extended frames. Furthermore, it is possible to use wildcards in certain attributes, e.g. the source address 1.*.* to match all frames coming from area 1.

As shown in Figure 3.1, each coupler is fitted with two interfaces: tp0 and tp1. In other filtering systems, it is possible to configure in and out rules. In rules for traffic coming into an interface, Out for traffic leaving an interface. It is generally considered good practice to filter traffic as soon as possible [Nor05]. Due to this, only traffic coming into an interface will be filtered, making all rules In.

There are three attributes to match the actual content of a frame: DPT, ObjectSize, Data. DPT refers to the datapoint types defined in DIN 50090 3-3. If the DPT attribute is set, the filter will verify that data being sent is a plausible value for the datapoint type. While

most communication takes place using standard datapoint types, this does not apply to all. To accommodate this, an object-size can be given as an alternative. It is also possible to match the exact content using regular expressions. Use-cases include limiting transmission to turning a device ON or limiting the transmitted temperature to a reasonable range of say, between -15°C to 30°C.

A formal definition of the rule language will be given as EBNF in Listing 3.1. The non-terminal symbol REGEX refers to any valid Perl Compatible Regular Expressions (PCRE). Rules are parsed top-to-bottom. As soon as the first rule is matched successfully, it does not compare the frame to any other rules.

3.5 Matching

Table 3.4 explains what criteria has to be met for an attribute to match a frame.

Parameter	Criteria
Src	If Src uses no wildcards, it must be identical to the frame's source address If Src uses a device wildcard, it must match the frame's area and line address If Src uses a line and device wildcard, it must match the frames area address
DestType	If DestType is GRP, the address type must be set to 1 If DestType is IND, the address type must be set to 0
Dest (IND)	If Dest uses no wildcards, it must be identical to the frame's destination address If Dest uses a device wildcard, it must match the frame's area and line address If Dest uses a line and device wildcard, it must match the frames area address
Dest (GRP)	If Dest uses no wildcards, it must be identical to the frame's destination address If Dest uses a subgroup wildcard, it must match the frame's main and middle group If Dest uses a middle and subgroup wildcard, it must match the frame's main group
Data	The regular expression must match the frame's data
DPT	The DPTs regex must match the frame's data, see Section 3.8
ObjectSize	The transmitted data must match the length given in object size With respect to the fact, that data can only be transmitted in blocks of octets, it must be reassured that the remaining space is padded with zeros
FrameType	The frame type must be 1 if standard, 0 if extended
HopCount	The frame's hop count must be less or equal to the given value
Priority	The frame's priority must be less or equal to the given value
Service	The frame's service, encoded in the APCI field, must be an identical match

Table 3.4: Matching criteria for filter attributes

Rule	Description	Required	Example value
Policy	Course of action to take in case of positive matching	X	DROP
Src	Source address in frame	X	I.L.*
DestType	Type of destination address in frame	X	IND
Dest	Destination address in frame	X	I/*

Parameter	Description	Required	Example value
Data	Regular expression to match the frame's data cannot be used in combination with DPT or ObjectSize		/0000I/
DPT	Verify data looks plausible for given DPT cannot be used in combination with Data or ObjectSize		DPT1-1
ObjectSize	Verify data is not bigger than given ObjectSize cannot be used in combination with Data or DPT		8 bits
FrameType	Kind of data-frame		STANDARD
HopCount	Maximum HopCount allowed		6
Priority	Maximum priority allowed in frame		High
Service	Service used in frame		A_GroupValue_Write
establishedOnly	Allow certain services only after frame of another service was received. E.g. an A_GroupValue_Response only after an A_GroupValue_Read		-
log	Log frames matching this filter		-

Table 3.5: Structure of rules

<RULESET> ::= <RULE> | <RULE>, <RULESET>
 <RULE> ::= <POLICY>, <WS>, <SRC>, <WS>, <DEST>, <OPTIONS>, *EOL*
 <POLICY> ::= "FORWARD" | "DROP" | "NOISE"
 <SRC> ::= <PHYSICAL-ADDRESS>
 <DEST> ::= ("IND", <WS>, <PHYSICAL-ADDRESS>) | ("GRP", <WS>, <LOGICAL-ADDRESS>)
 <OPTIONS> ::= [<DataFilter>], [<FrameType>], [<HopCount>], [<Priority>], [<Service>], [<WS>, "--establishedOnly"], [<WS>, "--log"]
 <PHYSICAL-ADDRESS> ::= "*" | ((No-15), ".*", *) | ((No-15), ".*", *) | ((No-15), ".*", (No-15), ".*", (No-255))
 <LOGICAL-ADDRESS> ::= "*" | (No-65535) | ((No-31), "/**") | ((No-31), "/" | (No-2047)) | <LOGICAL-ADDRESS3>
 <LOGICAL-ADDRESS3> ::= ((No-31), "/", (No-7), "/**") | ((No-31), "/", (No-7), "/" | (No-255))
 <DataFilter> ::= <WS>, (<Data> | <DPT> | <ObjSize>)
 <Data> ::= "--data", <WS>, <REGEX>
 <DPT> ::= "--dpt", <WS>, ("DPST-1" | ...)
 <ObjSize> ::= "--objSize", <WS>, (<N>, "bits" | <N>, "bytes")
 <FrameType> ::= <WS>, "--frameType", <WS>, ("STANDARD" | "EXTENDED")

```

<HopCount>  = <WS>, "--hopCount", <WS>, <No-7>
<Priority>   = <WS>, "--priority", <WS>, ("System" | "Alert" | "High" | "Low")
<Service>   = <WS>, "--service", <WS>, ("A_GroupValue_Write" | "A_GroupValue_Read" | "A_GroupValue_Response" | ...)

<N>         = <DIGIT> | (<DIGIT>, <N>)
<No-7>      = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"
<No-15>     = ("0", <DIGIT>) | ("1", ("0" | "1" | "2" | "3" | "4" | "5"))
<No-31>     = (("0" | "1" | "2"), <DIGIT>) | ("3", ("0" | "1"))
<No-255>    = (("0" | "1"), <DIGIT>, <DIGIT>) | ("2", ("0" | "1" | "2" | "3" | "4"), <DIGIT>) | ("25", ("0" | "1" | "2" | "3" | "4" | "5"))
<No-2047>   = (("0" | "1"), <DIGIT>, <DIGIT>, <DIGIT>) | ("20", ("0" | "1" | "2" | "3"), <DIGIT>) | ("204", <No-7>)
<No-65535>  = (("0" | "1" | "2" | "3" | "4" | "5"), <DIGIT>, <DIGIT>, <DIGIT>, <DIGIT>) | (<No0000-65535>)
<No60000-65535> = ("6", ("0" | "1" | "2" | "3" | "4"), <DIGIT>, <DIGIT>) | (<No50000-65535>)
<No65000-65535> = ("65", ("0" | "1" | "2" | "3" | "4"), <DIGIT>, <DIGIT>) | ("655", ("0" | "1" | "2"), <DIGIT>) | (<No65530-65535>)
<No65530-65535> = ("6553", ("0" | "1" | "2" | "3" | "4" | "5"))
<DIGIT>     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<WS>        = " "
<REGEX>     = /* ANY PCRE-compatible VALUE */

```

Listing 3.1: EBNF of filter rules

3.6 State

In order to provide additional filter options like `Established only` or rate-limiting for preventing abuse, it is essential to keep track of previous frames and build up an internal representation of the state. In order to do this, it is crucial to have two kinds of information. A combination of attributes that build a unique key to identify the communication and a number of attributes to describe the current state. The unique key differs from communication mode to communication mode. In a simple point-to-multipoint (multicast) communication, a set of communications can be uniquely identified by its group-address. Broadcast connections, including point-to-domain and point-to-all-points, can be described using the originator's source address. Point-to-point communication can be identified by combining the addresses of both participants.

The representation of the current state depends on the used communication method. Exemplary for point-to-multipoint communication, the state can be described using two attributes: The communication-object's value, , that can be taken from frames with the services `A_GroupValue_Response` and `A_GroupValue_Write`, and the last invocations of a `A_GroupValue_Write` frame.

In the context of point-to-multipoint communication, the keyword `Established only` can be used for rules matching the service `A_GroupValue_Response`. Using that keyword will ensure, that the group-object has been previously polled by a `A_GroupValue_Read` request.

3.7 Generating rules

As outlined in chapter 2 “Background and prior work”, ordinary traffic will exclusively take place using point-to-multicast communication, relying on three services:

- A_GroupValue_Read
- A_GroupValue_Response
- A_GroupValue_Write

The requirement is to fully restrict frames to the bare minimum, allowing only frames that are absolutely necessary in order to keep the buildings functionality working. To fulfil this requirement, a whitelist approach was chosen. Frames, that need to be transmitted to other lines, will be forwarded. Frames, that are only required on the local line will be dropped by the router, and frames, that are not expected at all, will be interrupted by noise.

To further precise filter rules, it is necessary to extract data from ETS. Data that proves advantageous includes:

1. Maximum priority
2. Communication flags
3. sending group-addresses
4. Datapoint type (or object size)

The first and fourth are obvious, having the ability to directly map them to attributes in the filter language. The communication flags of a device object can be used to specify the services, that may be used in combination of the device’s physical address and the communication object’s group address. Each communication flag is identified with certain services. For reference, a list of communication flags was given in Chapter 2 in Table 2.6. To recall, every device object can be associated with multiple group addresses, but at most with one sending group-address.

A device can have the multiple device objects referring to the same group-address. In order to prevent having multiple rules for the same combination of source address and

group address, device objects for the same device address and group address need to be merged. The merger has the biggest set of permissions: E.g. One device with “transmit-enabled” set to true, the other with false, the merger’s “transmit-enabled” flag will be set to true. With regard to the priority, the maximum priority has to be chosen.

Device objects, whose “communication enable” flag is set to false, shall be ignored completely and will not be allowed to communicate.

If the “transmit enable” flag of a device object is set to true, the device shall be allowed to send `A_GroupValue_Write` requests, using only it is sending group-address. If there are devices on other lines, listening to that group-address, the router shall forward the frame to the superordinate line. Otherwise, it shall simply be dropped, but not be interrupted by noise.

If a device object’s “write-enabled” flag is set to true, it expects incoming `A_GroupValue_Write` requests. Frames from other lines, that are sent to any of the device object’s associated group-addresses, need to be forwarded by the router to its subordinate line.

Devices with a “read-enabled” flag can only be used in combination with devices with an “update-enabled” flag. Devices with an enabled “update-enabled” flag will send out `A_GroupValue_Read` requests, whenever they deem necessary and react the to the corresponding `A_GroupValue_Response` request by updating their internal value. Devices with an enabled “read-enabled” flag are expecting incoming `A_GroupValue_Read` requests on any of their associated group-addresses and will respond with an `A_GroupValue_Response` frame using their sending group-address.

Chapter 2 further outlined, that a frame, having its hop count to seven, will be indefinitely forwarded, ignoring the amount of routers it passes. It was said, that this behaviour is only desired for management communication and is not used for ordinary group-communication. This is why the filter rules will have their hop count attribute set to six.

This was implemented in algorithm 1 and shall be briefly explained as follows. Filter rules are generated on a per router basis. Line 2 and 3 create empty sets for ingress and egress rules. Afterwards, all group-addresses associated with devices in the subordinate line

will be gathered. For each group-address, a list of associated devices is fetched and sorted by their position in the network, by their enabled communication flags, and if necessary, the devices are filtered to only match their sending group-address. Lines 30 to 69 are responsible for generating the actual rules per the ideas, described above. Fallback-rules are added on line 72 and line 73. As a router is only supposed to send noise to its subordinate line, the fallback rule using noise is only added to the set of egress rules. Unexpected frames coming from the superordinate line, will simply be dropped.

The implementation of this algorithm will be discussed in Chapter 4.1 “Generating rules from ETS”, and is provided in the appendix A “Disc with supplementary material”.

Depending on the router selected, the amount of rules can grow rather big. The basis of the following numbers is a set of rules generated for all 43 routers of the computer science building of the University of Rostock, a mid-size building with four floors. The number of rules is given in table 3.6. The router with the biggest ruleset, containing a total amount of 2848 rules, 2404 ingress rules and 444 egress rules, is router 1.0.0. Nevertheless, Figure 3.2 shows that routers with more than 1500 rules are the exception rather than the rule. The performance impact for a router, having to deal with 2848 rules, has to be tested in an experiment.

Description	Number of rules
Total number of rules on all routers	32866
Mean	764
Median	725
Standard deviation	728.74
Min	2
Max	2848

Table 3.6: Size of ruleset for computer science building



Figure 3.2: Number plot of ruleset size, one dot represents the amount of rules for one router

Algorithm 1 Filter rule generator (part 1/3)

```

1: procedure GENERATERULES(area, line)
2:   ingress  $\leftarrow$  []
3:   egress  $\leftarrow$  []
4:
5:   if line = 0 then                                     ▷ If line is zero, this is a main line
6:     groupAddresses  $\leftarrow$  getGroupAddressesForArea(area)
7:   else
8:     groupAddresses  $\leftarrow$  getGroupAddressesForLine(area, line)
9:   end if
10:
11:  for all groupAddress in groupAddresses do
12:    devices  $\leftarrow$  getAllDevicesForGroupAddress(groupAddress)
13:    devices  $\leftarrow$  filterForActiveDevices(devices)
14:    datapointtype  $\leftarrow$  getDataPointTypeForGroupAddress(groupAddress)
15:    objectsize  $\leftarrow$  getObjectSizeForGroupAddress(groupAddress)
16:
17:    internalDevices  $\leftarrow$  filterDevicesInsideCoupler(devices, area, line)
18:    externalDevices  $\leftarrow$  filterDevicesOutsideCoupler(devices, area, line)
19:
20:    internalReadAll  $\leftarrow$  filterReadFlag(internalDevices)
21:    internalReadSendOnly  $\leftarrow$ 
22:      filterSending(filterReadFlag(internalDevices))
23:    internalTransmitSendOnly  $\leftarrow$ 
24:      filterSending(filterTransmitFlag(internalDevices))
25:    internalUpdateAll  $\leftarrow$  filterUpdateFlag(internalDevices)
26:    internalWriteAll  $\leftarrow$  filterWriteFlag(internalDevices)
27:    externalReadAll  $\leftarrow$  filterReadFlag(externalDevices)
28:    externalReadSendOnly  $\leftarrow$ 
29:      filterSending(filterReadFlag(externalDevices))
30:    externalTransmitSendOnly  $\leftarrow$ 
31:      filterSending(filterTransmitFlag(externalDevices))
32:    externalUpdateAll  $\leftarrow$  filterUpdateFlag(externalDevices)
33:    externalWriteAll  $\leftarrow$  filterWriteFlag(externalDevices)

```

Algorithm 1 Filter rule generator (part 2/3)

```

30:     if count(internalWriteAll) > 0 then
31:         for all d in externalTransmitSendOnly do
32:             ingress[] ← "FORWARD d.physicalAddr GRP groupAddr \
--frameType STANDARD --service A_GroupValue_Write \
--priority d.priority --hopCount 6 --dpt d.DPT"
33:         end for
34:     end if
35:
36:     for all d in internalTransmitSendOnly do
37:         if count(externalWriteAll) > 0 then
38:             egress[] ← "FORWARD d.physicalAddr GRP groupAddr \
--frameType STANDARD --service A_GroupValue_Write \
--priority d.priority --hopCount 6 --dpt d.DPT"
39:         else
40:             egress[] ← "DROP d.physicalAddr GRP groupAddr \
--frameType STANDARD --service A_GroupValue_Write \
--priority d.priority --hopCount 6 --dpt d.DPT"
41:         end if
42:     end for
43:
44:     if count(internalUpdateAll) > 0 then
45:         for all d in externalReadSendOnly do
46:             ingress[] ← "FORWARD d.physicalAddr GRP groupAddr \
--frameType STANDARD --service A_GroupValue_Response \
--priority d.priority --hopCount 6 --dpt d.DPT \
--establishedOnly"
47:         end for
48:     end if
49:
50:     if count(internalReadAll) > 0 then
51:         for all d in externalUpdateAll do
52:             ingress[] ← "FORWARD d.physicalAddr GRP groupAddr \
--frameType STANDARD --service A_GroupValue_Read \
--priority d.priority --hopCount 6 --data ^O{6}$"
53:         end for
54:     end if

```

Algorithm 1 Filter rule generator (part 3/3)

```

55:   for all d in internalUpdateAll do
56:     if count(externalReadAll) > 0 then
57:       egress[] ← "FORWARD d.physicalAddr GRP groupAddr \
                    --frameType STANDARD --service A_GroupValue_Read \
                    --priority d.priority --hopCount 6 --data ^O{6}$"
                    --establishedOnly"
58:     else if count(internalReadAll) > 0 then
59:       egress[] ← "FORWARD d.physicalAddr GRP groupAddr \
                    --frameType STANDARD --service A_GroupValue_Read \
                    --priority d.priority --hopCount 6 --data ^O{6}$"
60:     end if
61:   end for
62:
63:   for all d in internalReadSendOnly do
64:     if count(externalUpdateAll) > 0 then
65:       egress[] ← "FORWARD d.physicalAddr GRP groupAddr \
                    --frameType STANDARD --service A_GroupValue_Response \
                    --priority d.priority --hopCount 6 --dpt d.DPT \
                    --establishedOnly"
66:     else if count(internalUpdateAll) > 0 then
67:       egress[] ← "DROP d.physicalAddr GRP groupAddr \
                    --frameType STANDARD --service A_GroupValue_Response \
                    --priority d.priority --hopCount 6 --dpt d.DPT \
                    --establishedOnly"
68:     end if
69:   end for
70: end for
71:                                     ▷ Add fallback rules
72: ingress[] ← "DROP any any any"
73: egress[] ← "NOISE any any any --log"
74:
75:   return ingress, egress
76: end procedure

```

3.9 Example

As an example, the KNX-experimentation cupboard from the university's lab 308 was taken. A visualisation of the network structure is shown in Figure 3.4. The network consists of three lines, one of which is an area line. Two line couplers connect line 1.1 and 1.2 to the area line.

Line 1.1 contains three devices with an individual address. A LED-Controller with the address 1.1.1, an 8-times push button with the address 1.1.21 and a power supply with the address 1.1.254. Line 1.2 contains only one device with an individual address, a presence detector with the address 1.2.21. As a side note, line 1.2 is evidently also fitted with a power supply, it was just not assigned its own physical address. The area line does not contain any other device besides the line couplers.

The network contains 5 different group addresses: 1/1/1 through 1/1/5. These group addresses are assigned to three different devices. The configured communication flags can be seen in Figure 3.4. C corresponds to the “communication-enable”, R to the “read-enable”, W to the “write-enable” and T to the “transmit-enable” flag. The resulting ruleset is provided in table 3.7.

Despite some devices having an enabled “read-enable” flag, there are no rules matching the services `A_GroupValue_Read` or `A_GroupValue_Response`, because there are no counterpart devices with enabled “update-enable” flag. The coupler 1.1.0 drops all egress frames, because there are no devices with “write-enable” flag outside the line 1.1.

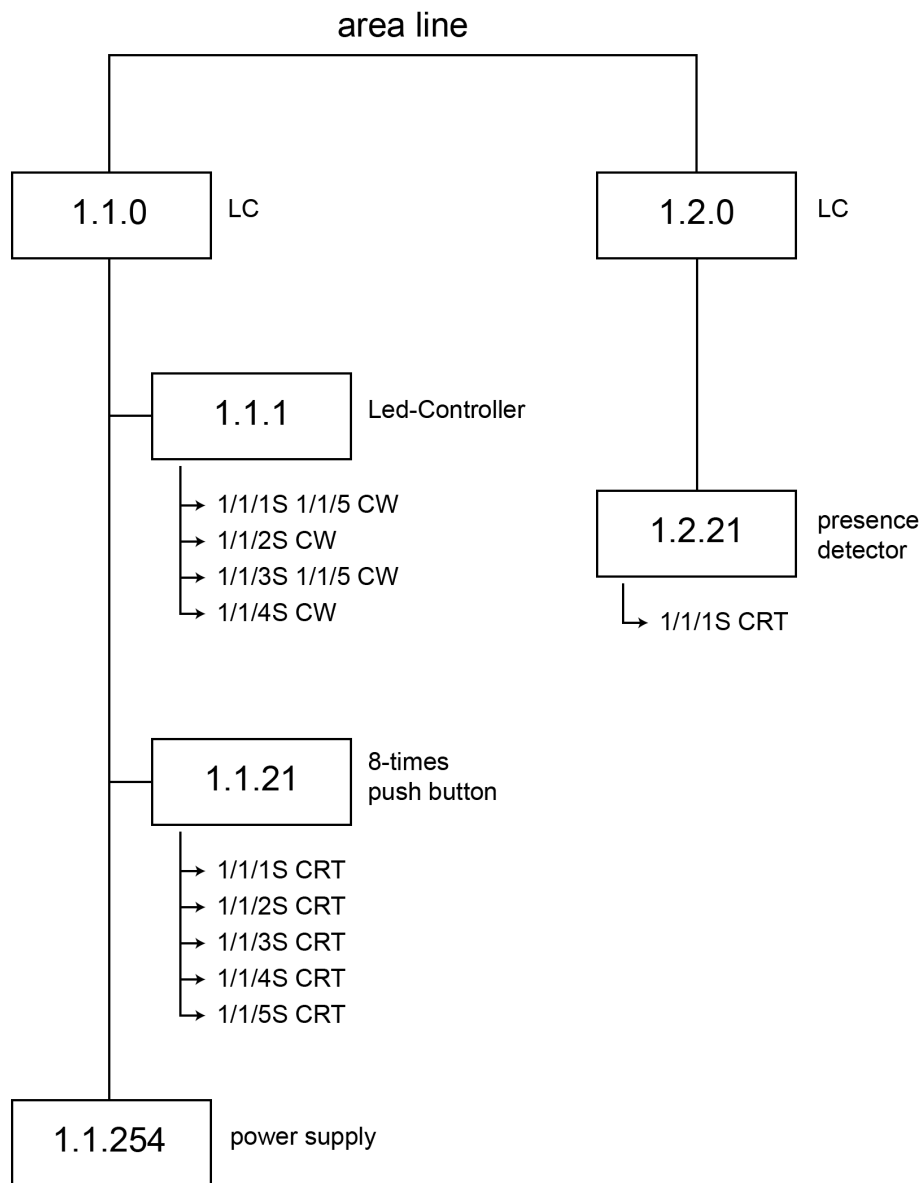


Figure 3.4: Structure of “Experimentalkasten KNX” in lab R308

Coupler Address	Filter	Policy	Src	DestType	Dest	Options	
1.1.0	Egress	DROP	1.1.21	GRP	1/1/1	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-1	
		DROP	1.1.21	GRP	1/1/2	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-3	
		DROP	1.1.21	GRP	1/1/3	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-1	
		DROP	1.1.21	GRP	1/1/4	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-3	
		DROP	1.1.21	GRP	1/1/5	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-1	
	Ingress	NOISE	any	any	any	any	--log
		FORWARD	1.2.21	GRP	1/1/1	1/1/1	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-1
		DROP	any	any	any	any	
		FORWARD	1.2.21	GRP	1/1/1	1/1/1	--frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6 --dpt DPST-1
		NOISE	any	any	any	any	--log
1.2.0	Ingress	DROP	any	any	any		

Table 3.7: Ruleset for “Experimentalkasten KNX” in lab R308

3.10 Attacks

Background and prior work discussed general attacks on BAS as well as KNX security issues in detail. Figure 2.1, introduced in “Security in Building Automation Systems” showed a categorisation of attacks. On its highest level, it distinguishes between network attacks and device attacks.

Network attacks is separated into interception, fabrication, modification and interruption. The proposed filtering system is conceptually not capable of solving interception attacks, because it is not possible to tell whether or not an additional device is listening on the bus. Furthermore Thomas Mundt showed in his publication “Security in building automation systems - a first analysis” that one does not even require direct physical access to the bus: “ In a second attempt we were also able to read from the KNX bus without touching any cable. We tried several coils to receive suitable impulses from the cables within a wall. [...] An audio pre-amplifier and a notebook for recording proved to be more than suitable for our purpose. [...] Utilizing audio equipment lowers the costs for an attacker dramatically.” [MW16, p. 6]

While replay attacks and inserting legitimate looking frames is not prevented, their functionality is limited to that of the existing devices on the line. For example: Given three lines, one line with a push button and two lines with a light actor each, being configured with different group-addresses. The push button on line one is configured to switch or dimm the light on line two. An attacker on line one will be able to send switch or dimm commands to the light on line two, but he won't be able to suddenly control the light actor on line three.

The same goes for Man-in-the-middle attacks. While alteration of the sent value is generally possible, the attacker cannot change the frame's destination address to one the original device was never supposed to communicate with. The capabilities of the attacker are limited to those of the devices already on the line.

Interruption attempts like Denial of Service or network flooding are being limited to the local line, since these frames will not be forwarded by the router. The other category

of attacks, being device attacks, is further sorted into software, side-channel and physical. The filtering system will prevent any kind of management communication from unauthorised entities. An attacker won't be able to set configuration flags or send special requests like `DM_Restart` (see *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 7-1: Systemmanagement - Managementverfahren; Deutsche Fassung EN 50090-7-1:2004, Text Englisch*). All this does not guarantee, that an attacker cannot exploit existing security issues in the software of the device itself. Side-channel and physical attacks cannot be prevented. Electricians need to take precautions and prevent any physical access to the bus system and KNX devices. Best practices are outlined in *KNX Sicherheit - KNX Positionspapier zu Datensicherheit und Datenschutz*. Nevertheless, even in the case of entire devices being replaced, their functionality will be limited to those of other devices on the line.

3.11 Rule parsing

With a growing amount of rules, matching the frame against all rules takes a considerable amount of time. There are various approaches for faster matching discussed below.

Different approaches are required when the rules have interdependencies, meaning the rule order does not only have performance repercussions, but a different rule order might also result in a different policy decision. Because the rule-generator, proposed earlier in this chapter, only creates rules without interdependencies, having either one rule matching the frame or falling back to the default rules, the topic of interdependencies will be omitted. Optimisation, with respect to redundancies in the context of wildcards will be disregarded because such rules will never occur with the proposed rule-generator.

The first approach is keeping the list unordered. It is the easiest, requiring no additional work. On the downside, the order will solely depend on the order the rules were outputted by the generator tool, having absolutely no relation to the amount of data or frequency of frames coming into the router.

Another approach is putting frequently matched frames first. It comes in different variations. It is possible to sort the list once based on sample data or to do it continuously, as

the router is in use. Depending on the environment, it might also be beneficial to introduce different time-slots, having a different list order at different hours. The downside to this method is that it requires more work and resulting in an especially higher system load if it is done continuously.

The last approach is putting the last matched rule on top. This comes with the drawback that it requires a write operation after every match, given the frame did not match the first rule in the list. It is deemed useful for lines, where there is only one dominant device like a light switch, but has a foreseeably inferior performance on lines with lots of different traffic, especially area lines or backbone lines

Implementation

Contents

4.1	Generating rules from ETS . . .	62
4.2	Remarks on implementation of ECU	63

This chapter covers the topic of implementation. In the first section 4.1 “Generating rules from ETS”, the implementation of the rule generator is discussed. It also elaborates on particularities of the ETS database structure, that were of significance for the correct implementation of the algorithm. Subsequently, section 4.2 gives short remarks on the implementation of the ECU.

4.1 Generating rules from ETS

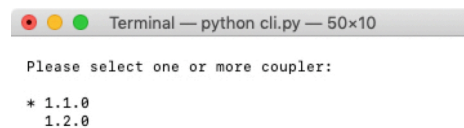
The rule generator was implemented as a Python 3 tool, consisting of two modules: *cli.py* and *gen.py*. *gen.py* contains the main logic, following the algorithm proposed in 3.7 “Generating rules”. It extracts all the information from ETS’s MsSQL database. The main function is `generate_rules_for_coupler`. It takes four parameters: `conn`, the active database connection, and `installation_id`, `area`, `line` to identify the router. The very core of *gen.py* is `get_all_devices_related_to_group_address`. It queries all the devices related to a group-address, using a database join of the different tables - `Connector`, `GroupAddress`, `Device`, `Line`, `Area`, `DeviceObject` and `CommunicationObjectRef`. Configuration flags are stored in a hierarchic structure. Device-object specific configurations are stored in the `DeviceObject` table. If no value is given, the value is inherited from `ConfigurationObjectRef`, respectively `ConfigurationObject`. To simplify the query, it relies on stored procedures to fetch configuration flags, if they are not set in the table `DeviceObject`, and to format the group-address in the human-readable format. In the database, group-addresses are simply stored as the integer representation of the group-address, as it is sent on the bus.

cli.py provides an interactive command line interface. It takes five different parameters: `--dbhost`, `--dbuser`, `--dbpass`, `--dbname`, `--filename`. They can be provided as command line arguments or will alternatively be asked for in the interactive command line interface. After taking all the arguments and connecting successfully to the MsSQL database, it queries all projects and lists them to the user, as seen in Figure 4.1. The user has to select one project. Upon selection, the program will query all couplers of a project and list them. To get all couplers, it queries all devices with a join on the `Area` and `Line` tables, filtering for all devices with the device-address being set to zero. The user can select multiple couplers. *cli.py* will call the routines from *gen.py* for each selected coupler, writing them into one file, grouped by coupler.



```
Terminal — python cli.py — 50x10
Please select a project:
ITMZ 9KI606
> Show-Case
```

Figure 4.1: Select project screen



```
Terminal — python cli.py — 50x10
Please select one or more coupler:
* 1.1.0
1.2.0
```

Figure 4.2: Select coupler screen

```

Terminal — less rules.txt — 100x10
# Rules for 1.1.0
## Egress rules
DROP 1.1.21 GRP 1/1/1 --frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6
--dpt DPST-1
DROP 1.1.21 GRP 1/1/2 --frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6
--dpt DPST-3
DROP 1.1.21 GRP 1/1/3 --frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6
--dpt DPST-1
DROP 1.1.21 GRP 1/1/4 --frameType STANDARD --service A_GroupValue_Write --priority Low --hopCount 6
rules.txt

```

Figure 4.3: Output file of cli.py

The result will be stored in the file given in `--filename`, provided by the user as argument. `cli.py` has different python dependencies: `click`, providing utility functions for building interactive cli applications, `pymssql` as a connector to the MsSQL database, and `pick` gives a ready-to-use implementation of the multi-select list.

4.2 Remarks on implementation of ECU

The Extended Coupler Unit should be implemented as a bus device with two interfaces. It has to be connected to the bus directly and cannot be connected via KNX-to-USB interfaces, because they don't allow sending noise. Furthermore, KNX-to-USB interfaces will only forward frames as the reception of the frame on the bus is complete. As outlined in Chapter 3, it is required to match the frame as it is still coming in, to be able to prevent transmission by sending noise.

Filter rules should be converted to regular expressions and combined to one automata, as described in Chapter 2.4 “Deep Packet Inspection”.

The state table needs various columns: The communication-mode to differ between point-to-point, point-to-broadcast or point-to-multicast connections. Furthermore, it needs columns for all the attributes discussed in section 3.6 “State”

CHAPTER 4. IMPLEMENTATION

Evaluation

Contents

5.1	Methodology	66
5.2	Assessing Results	68
5.3	Thought experiment	69

This chapter proposes a methodology for evaluation. The first subsection 5.1 “Methodology”, answers different questions including what to evaluate, how to evaluate it and how to gather the data for testing. Subsequently, section 5.2 “Assessing Results”, deals with the subject of rating the results.

5.1 Methodology

To evaluate the efficacy of the proposed concept, it is necessary to cover maximum ground:

1. Is the proposed filter system capable of preventing attacks? If yes:
 - (a) What kind of attacks does it prevent?
2. What is the rate of false positives?
3. What is the rate of false negatives?
4. Is it reducing the general load on lines?

The last point is of special interest, because the proposed filter system replaces that of a standard KNX router completely. At its worst, a broken filtering system would result in all traffic being forwarded to every line, drastically increasing load on the entire system. The evaluation will be realised as an experiment. To answer the above-noted questions, it is necessary to gather two data sets. The first set of data, hereinafter called real-world-data, is assumed to be free of attacks and has to be captured from a KNX system. The second data set contains artificially crafted frames that are known to embody attacks.

For the purpose of collecting real-world-data, the data shall be captured on a main line in the third floor of the computer science building of University of Rostock. The main line and every subline (not the backbone line) shall be fitted with sensors. The data on these lines shall be captured for a continuous period of two weeks. While it is not strictly necessary, it should be preferred to conduct the data capturing during lecture time, in order to have a bigger sample size. Before recording the data, it is necessary to inform and request the consent of all the employees, whose offices happen to be connected to any of the examined lines. The captured data will be grouped by line. Consequently, frames that were forwarded to other lines, will be recorded individually for each line they occurred on. Frames need to be flagged whether they were sent by a device on this line or on another line. This can simply be done by inspecting the source address. For each frame, it shall also be noted, whether a device on that line replied with an acknowledgement frame. This gives an indication as to whether this line contains devices associated

with the frame's destination address. Area lines and backbone lines shall be marked, when an acknowledgement frame was sent on any of their subordinate lines.

To craft artificial frames for the second set, it is essential to compile a list of attacks to test against. The drawback to this approach is that it cannot be tested against unknown attacks. Nonetheless, the approach is considered satisfactory. The list will be taken from the prior research presented in chapter 2 "Background and prior work". (Fig. 2.1).

As explained prior to this chapter, it is not possible to tell whether an attacker is eavesdropping on the bus. Hence it is implausible to craft frames representing interception attacks. For replay attacks, a small subset of the captured real-world-data shall be selected, their time-stamps altered and the frames shall be added to the set of forged-data. With regards to inserting messages, frames shall be crafted in a way, that they revert an action from frames present in the real-world-data set. The time-stamp of the crafted frame shall be set to a few milliseconds later than that of the real-world-data frame. Different techniques to Denial of Service (DoS) attacks include shorting the line, flooding the bus with legitimate requests and flooding the bus with random data. Since the router will ever only forward actual frames and no noise, the crafted frame shall be a legitimate request. To cause the maximum damage, this frame shall use the service `A_Restart`, which causes devices to restart and be unavailable for some seconds. These `A_Restart` frames shall be sent to both devices on the same line as the attacker, but also to devices on other lines. Reconfiguration attacks can be crafted using the services `A_UserMemory_Write` and `A_UserMemory_Read`, altering and reading a device's configuration. Reconnoitring attacks, that are solely based on network sniffing, cannot be recognised. To cover reconnoitring attacks that are based on pinging individual devices, the set of forged-data shall be extended to contain frames using the service `A_IndividualAddress_Read`, reading out the individual address of each device, but also frames using the service `A_Connect`, trying to establish a connection-oriented point-to-point communication.

Based on the existing two sets of real-world-data and forged-data, two new sets will be created:

- The test-data-set: A merger of real-world-data and forged-data, sorted by line, filtered for frames sent on the line in question and flagged by the original set of either real-world-data or forged-data

- The desired-data-set: The set of real-world-data without the frames that were not flagged as being acknowledged.

The selected subnetwork, that data was captured on in the first step, needs to be implemented in a software simulation, including software-based extended coupler units. For each ECU, a most restrictive set of filter rules shall be generated using the tool developed in this master thesis. Similar to the actual network, each line in the simulation has to be fitted with a sensor.

To conduct the actual experiment, the captured data in the test-data-set needs to be replayed in the simulator, each frame originating from its original line and at its original timestamp. The data captured at the virtual sensors will be stored in a simulation-data-set.

5.2 Assessing Results

For assessing the experiment's result, each line of the simulation data-set must be compared with that of the desired data-set.

If the two sets are identical, all the filters have functioned as was expected of them and all attacks have been prevented.

In case the two sets are not identical, they must be compared on a line and frame basis:

- A frame, that is part of the desired-data-set, but not the simulation-data-set, was erroneously filtered and needs to be considered a false positive.
- If a frame is part of the simulation-data-set, but not the desired-data set, it is necessary to see what set the frame came from originally.
 - real-world-data: The frame was forwarded to a line, where no one expects it. This does not pose a threat, but creates unnecessary traffic.
 - forged-data: This frame is a false negative. The attack was not captured by the filter system.

5.3 Thought experiment

Imagine the following scenario: A KNX network contains 4 different areas, all connected by a common backbone line. The areas are numbered 1.0.0, 2.0.0, 3.0.0 and 4.0.0. The fictional building has three floors. Each floor is associated with one area. The area 4.0.0 is used for devices on the building's roof. All floors are fitted with 15 lines. Line 1 is representing the hallway, with 4 presence detectors and a light. The lines 2 to 15 of each area represent individual rooms with a 4-times push button, a light and shades. The roof, area 4.0.0, is only fitted with one line, containing a sunlight detector. A total amount of four backbone couplers and 46 line couplers are used in this network. The complete formation is visualised in Figure 5.1.

Table 5.1 shows the estimated amount of frames per day for a device-type. A KNX system, using default couplers without filters, will cause traffic to be forwarded to each line, giving a total amount of 33.394 frames per line per day, totalling up to 12.188.810 frames per line per year. By replacing the KNX coupler with an ECU using the automatically generated rules, the amount of traffic could drastically be reduced.

For an average room, the traffic would be reduced to 297.110 per year. This is a reduction of 97,56%. The difference is even bigger in a hallway, reducing the amount of frames to 56.210 per year, a reduction of 99,54%. The biggest difference can be measured on the backbone line. When using the automatically generated set of filters, the traffic can be reduced to 1.460 frames. That's 99,99% less compared to before.

Device-type	Estimated Frames per day
Presence detectors	150
4-times push button - switching light	10
4-times push button - dimming light	500
4-times push button - shades	300
sun light detector	4

Table 5.1: Estimated amount of frames per day per device

CHAPTER 5. EVALUATION

The sunlight detector shall be exemplarily analysed using the Goltz-factor, because it is a significant device, being able to cause big damage. The Goltz-factor of the two scenarios, the first with no filtering at all and the second with an ECU, are compared in Table 5.2

The Goltz-factor of the sunlight-sensor is almost 80%, when using no filtering at all, but could be brought down to 41,25%, by replacing existing filters with an Extended Coupler Unit.

Based on this Goltz-factor, it is possible to answer the first evaluation question “Is the proposed filter system capable of preventing attacks” with yes. Furthermore, question 4 “Is it reducing the general load on lines?” can be answered with a yes as well, respecting the findings on the previous page. This thought experiment does not provide the means to answer question 1 (a), 2 and 3.

Class	No filtering	ECU
Device-class	3	3
Access class device	0	0
Access class bus	1	1
Reachability by other devices	2	1
static Goltz score	72,65%	58,33%
Amount of frames	2	0
Kind of frames	1	0
dynamic Goltz score	84,98%	0%
total Goltz score	79,06%	41,25%

Table 5.2: Value for different Goltz-classes / the Goltz-score

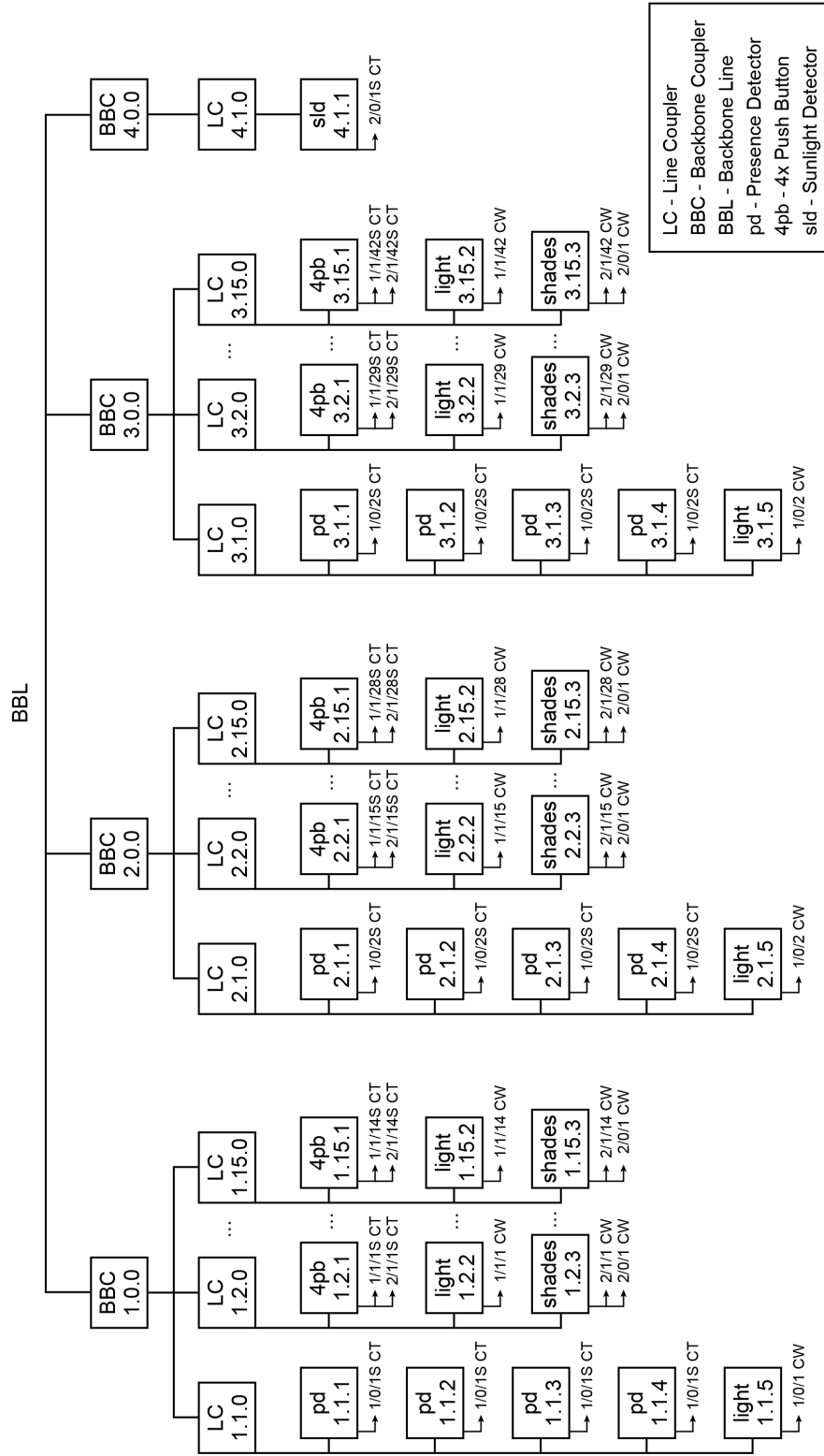


Figure 5.1: Fictional KNX network structure

CHAPTER 5. EVALUATION

Conclusion

This master thesis introduced the Extended Coupler Unit, a replacement for existing KNX couplers. It includes an enhanced filtering system, allowing deep packet inspection methods to be applied to KNX. A method for analysing data on the bus was provided and exemplary applied to KNX/TP1. The same method is applicable to other transmission media. It was analysed, which data is in the software surrounding the KNX ecosystem. ECUs provide stateful filtering, based on a novel language for filtering rules, that's human-readable and easy to maintain. While this system was developed with KNX in mind, the same approach should be applicable to other field bus systems as well.

A new algorithm was developed, that automatically generates a set of most restrictive filter rules based on the data extracted from ETS. It derives the permitted set of services from the configuration flags and builds rules based on service, priority, datapoint type and frame type.

With a growing set of rules, matching will take a considerable longer time. To optimise rule parsing, different approaches have been discussed, including their benefits and drawbacks.

A theoretical analysis of what attacks should be prevented was given. Nonetheless, this should be tested with the evaluation method provided in Chapter 5 "Evaluation".

CHAPTER 6. CONCLUSION

A method for evaluation was introduced. It is based on a mix of real-world-data and forged data containing attacks. All aspects like how to gather the real-world-data, how to create the forged data and how to perform the actual experiment have been discussed. Furthermore it was discussed how to rate the results of the experiment. Due to time constraints it was not possible to perform the evaluation. To provide some kind of evaluation nonetheless, a thought experiment was conducted, showing very promising results.

Future work

Due to time constraints, it was not possible to perform the full evaluation, as described per methodology in section 5.1 “Methodology”. This evaluation should be done. The results should be rated based on the system provided in 5.2 “Assessing Results”.

It should be researched, whether it is possible to extract more precise filter rules from ETS. This could be done by device type, e.g. a motion sensor may only invoke “Light On”, but not “Light Off”. It should be analysed, if this information can also be extracted from the application data, downloaded by ETS from the vendor’s website.

For the matter of this thesis, KNX Data Secure and KNX IP Secure was considered out of scope. As it is to be expected, that KNX Data Secure and KNX IP Secure will gain wider acceptance in the future, the concept should be enhanced to cover these KNX extensions, possibly with decrypting the data for analysis on the ECU.

One possible topic of investigation is the further optimisation of the rule-generator. It should be examined, whether it is possible to automatically combine rules, e.g. replacing multiple DROP rules with one DROP rule and a wildcard-identifier.

Another enhancement would be time-based filtering, e.g. only allow the sunlight sensor to send data between sunrise and sunset. This could also be used to apply logging only for certain hours, e.g. logging data from motion sensor between 1am and 5am.

CHAPTER 7. FUTURE WORK

List of Figures

1.1	Heatmap of internet-accessible KNX Gateways	2
1.2	Heatmap of internet-accessible BACnet Gateways	2
1.3	Categorisation of attacks on BAS	3
2.1	Categorisation of attacks on BAS	7
2.2	The ISO/OSI reference model	9
2.3	OSI Layers of KNX	9
2.4	Physical segments of a KNX/TP1 network in different arrangements	11
2.5	Relation of digital signal and serial bit stream	12
2.6	An octet mapped to a serial character	14
2.7	Visualised maximum buildout of a KNX network	16
2.8	Example topology with physical and logical addresses	18
2.9	Structure of a group object	19
2.10	Structure of standard data frame	21
2.11	Structure of extended data frame	21
2.12	Structure of poll frame	23
2.13	Response scheme of poll frames	23
2.14	Structure of acknowledgement frame	24
2.15	Visualisation of functional blocks	26
2.16	Example of the KNX interworking model	27
2.17	Standard representation of functional blocks	27
2.18	Datapoint type for switching	28
2.19	Datapoint type for dimming	28

LIST OF FIGURES

3.1	Simple illustration of a coupler	37
3.2	Number plot of ruleset size, one dot represents the amount of rules for one router	50
3.3	Datapoint type definition for DPT 1.001 - switching	54
3.4	Structure of “Experimentalkasten KNX” in lab R308	56
4.1	Select project screen	62
4.2	Select coupler screen	62
4.3	Output file of cli.py	63
5.1	Fictional KNX network structure	71

List of Tables

2.1	Comparison of KNX/TP1-64 and KNX/TP1-256	11
2.2	Maximum size of a KNX network in a KNX/TP1-64 environment	16
2.3	Structure of physical address	18
2.4	Structure of group addresses with 2 levels	18
2.5	Structure of group addresses with 3 levels	18
2.6	Meaning of configuration flags	19
2.7	Group object related application layer services	22
3.1	Different courses of action for a frame	38
3.2	Metadata of frames	39
3.3	Data extractable from ETS	41
3.4	Matching criteria for filter attributes	43
3.5	Structure of rules	44
3.6	Size of ruleset for computer science building	50
3.7	Ruleset for “Experimentalkasten KNX” in lab R308	57
5.1	Estimated amount of frames per day per device	69
5.2	Value for different Goltz-classes / the Goltz-score	70

LIST OF TABLES

List of Acronyms

ACL Access Control list	31
AES Advanced Encryption Standard	34
APCI Application layer Protocol Control Information	20
BAS Building Automation system	2
BCI BatiBUS Club International	8
CSMA/CA Carrier Sense Multiple Access / Collision Avoidance	13
CSMA/CD Carrier Sense Multiple Access / Collision Detection	13
DC Direct current	12
DES Data Encryption Standard	7
DFA Deterministic Finite Automata	32
DoS Denial of Service	67
EEPROM Electrically Erasable Programmable Read-Only-Memory	17
EFF Extended Frame Format	20
EHS European Home Systems	8
EHSA European Home Systems Association	8
EIB European Installation Bus	8

LIST OF ACRONYMS

EIBA European Installation Bus Association	8
ETS Engineering Tool Software	7
FDSK Factory Device Set up Key	29
HVAC Heating, Ventilation, Air Conditioning	3
LSB Least Significant Bit	14
MsSQL Microsoft SQL Server	40
NFA Non deterministic Finite Automata	32
PCRE Perl Compatible Regular Expressions	43
PSU Power Supply Unit	11
TPCI Transport layer Protocol Control Information	20

Bibliography

- [Ass] KNX Association. *KNX Sicherheit - KNX Positionspapier zu Datensicherheit und Datenschutz*. Last accessed on March 15th 2019. URL: https://www2.knx.org/media/docs/downloads/Marketing/Flyers/KNX-Secure-Position-Paper/KNX-Secure-Position-Paper_de.pdf (cit. on pp. 29, 59).
- [Ass13] KNX Association. *Grundlagenwissen zum KNX Standard*. Last accessed on March 15th 2019. 2013. URL: https://www.knx.ch/knx-chde/wdownload-d/Flyer/Endkunden/Grundlagenwissen_zum_KNX_Standard_German.pdf (cit. on pp. 8, 13, 15).
- [Bed10] Mark Bedner. “, , Deep Packet Inspection“ – Technologie und rechtliche Initiativen.” In: *Computer und Recht* 26.5 (Jan. 2010). DOI: 10.9785/ovs-cr-2010-339. URL: <https://doi.org/10.9785/ovs-cr-2010-339> (cit. on pp. 30, 32).
- [BJD16] J. Bugeja, A. Jacobsson, and P. Davidsson. “On Privacy and Security Challenges in Smart Connected Homes.” In: *2016 European Intelligence and Security Informatics Conference (EISIC)*. Aug. 2016, pp. 172–175. DOI: 10.1109/EISIC.2016.044 (cit. on p. 6).
- [Bra18] Thomas Brandstetter. *Securing industrial control systems - A peek into building automation security*. Last accessed on March 14th 2019. Mar. 2018. URL: <https://www.sans.org/webcasts/securing-industrial-control-systems-peek-building-automation-security-107275> (cit. on p. 3).

BIBLIOGRAPHY

- [Col17] M. Collins. *Network Security Through Data Analysis: From Data to Action*. O'Reilly Media, 2017. ISBN: 9781491962817 (cit. on pp. 9, 29–31, 36).
- [CS11] A. Chaudhary and A. Sardana. “Software Based Implementation Methodologies for Deep Packet Inspection.” In: *2011 International Conference on Information Science and Applications*. Apr. 2011, pp. 1–10. DOI: 10.1109/ICISA.2011.5772430 (cit. on p. 32).
- [DIN04a] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 3-2: Anwendungsaspekte - Anwendungsprozess ESHG Klasse 1; Deutsche Fassung EN 50090-3-2:2004, Text Englisch*. DIN. Sept. 2004. URL: <https://www.beuth.de/de/norm/din-en-50090-3-2/73716484> (cit. on p. 19).
- [DIN04b] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 4-1: Medienunabhängige Schicht - Anwendungsschicht für ESHG Klasse 1; Deutsche Fassung EN 50090-4-1:2004, Text Englisch*. DIN. June 2004. URL: <https://www.beuth.de/de/norm/din-en-50090-4-1/72067311> (cit. on p. 25).
- [DIN04c] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 4-2: Medienunabhängige Schicht - Transportschicht, Vermittlungsschicht und allgemeine Teile der Sicherungsschicht für ESHG Klasse 1; Deutsche Fassung EN 50090-4-2:2004, Text Englisch*. DIN. July 2004. URL: <https://www.beuth.de/de/norm/din-en-50090-4-2/72472562> (cit. on pp. 17, 24, 36).
- [DIN04d] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 5-2: Medien und medienabhängige Schichten - Netzwerk basierend auf ESHG Klasse 1, Zweidrahtleitungen (Twisted Pair); Deutsche Fassung EN 50090-5-2:2004, Text Englisch*. DIN. Sept. 2004. URL: <https://www.beuth.de/de/norm/din-en-50090-5-2/73718620> (cit. on pp. 10, 11, 13, 14, 20, 22).
- [DIN04e] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 7-1: Systemmanagement - Managementverfahren; Deutsche Fassung EN 50090-7-1:2004, Text Englisch*. DIN. Sept. 2004. URL: <https://www.beuth.de/de/norm/din-en-50090-7-1/73718965> (cit. on pp. 25, 59).
- [DIN09] DIN. *Elektrische Systemtechnik für Heim und Gebäude (ESHG) - Teil 3-3: Anwendungsaspekte - ESHG-Interworking-Modell und übliche ESHG-Datenformate; Englische Fassung EN 50090-3-3:2009*. DIN. Sept. 2009. URL: <https://www.beuth.de/de/norm/din-en-50090-3-3/120170190> (cit. on pp. 19, 26).

BIBLIOGRAPHY

- [Gol18] Johannes Goltz. *Sicherheitsanalyse von Gebäudeautomationsnetzen auf Feldbusebene am Beispiel von KNX*. Master thesis, University of Rostock, Faculty of Computer Science and Electrical Engineering, Institute for Computer Science, Chair for Information and Communication Services. 2018 (cit. on p. 34).
- [GPK10] W. Granzer, F. Praus, and W. Kastner. “Security in Building Automation Systems.” In: *IEEE Transactions on Industrial Electronics* 57.11 (Nov. 2010), pp. 3622–3630. ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2036033 (cit. on pp. 6, 7, 58).
- [Gra+06] W. Granzer, W. Kastner, G. Neuschwandtner, and F. Praus. “Security in networked building automation systems.” In: *2006 IEEE International Workshop on Factory Communication Systems*. June 2006, pp. 283–292. DOI: 10.1109/WFCS.2006.1704168 (cit. on pp. 6, 7, 33).
- [Hüb18] Christof Hübner. *Building Automation : Communication systems with EIB/KNX, LON and BACnet*. eng. 2nd ed. 2018. Signals and Communication Technology. 1 Online-Ressource (XII, 308 p. 216 illus., 13 illus. in color). Cham: Springer International Publishing, 2018. ISBN: 9783319732237 | 978-3-319-73223-7 | 9783319732220 (print) | 978-3-319-73222-0 (cit. on pp. 8, 10, 13–15, 17, 19, 24, 26).
- [Kas+05] W. Kastner, G. Neuschwandtner, S. Soucek, and H. M. Newman. “Communication systems for building automation and control.” In: *Proceedings of the IEEE* 93.6 (June 2005), pp. 1178–1203. ISSN: 0018-9219. DOI: 10.1109/JPRDC.2005.849726 (cit. on pp. 6, 8, 13, 15, 29).
- [Kum+06] Sailesh Kumar, Sarang Dharmapurikar, Fang Yu, Patrick Crowley, and Jonathan Turner. “Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection.” In: *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM ’06. Pisa, Italy: ACM, 2006, pp. 339–350. ISBN: 1-59593-308-5. DOI: 10.1145/1159913.1159952. URL: <http://doi.acm.org/10.1145/1159913.1159952> (cit. on p. 32).
- [KW06] Yuriy Kyselytsya and Thomas Weinzierl. “Implementation of the KNX Standard.” In: *KNXScientific Conference*. Last accessed on March 15th 2019. Oct. 2006. URL: https://weinzierl.de/images/download/references/KNX_Impl_E_2006_11_02.pdf (cit. on pp. 8, 10).

BIBLIOGRAPHY

- [Lou19] Vassilios Lourdas. *KNX Support: OPC-Export*. Last accessed on March 14th 2019. Feb. 2019. URL: <https://support.knx.org/hc/de/articles/115001823310-OPC-Export> (cit. on p. 40).
- [MKW12] T. Mundt, F. Krüger, and T. Wollenberg. "Who Refuses to Wash Hands? Privacy Issues in Modern House Installation Networks." In: *2012 Seventh International Conference on Broadband, Wireless Computing, Communication and Applications*. Nov. 2012, pp. 271–277. DOI: 10.1109/BWCCA.2012.51 (cit. on pp. 2, 10, 29).
- [Mun+11] A. Munoz, S. Sezer, D. Burns, and G. Douglas. "An Approach for Unifying Rule Based Deep Packet Inspection." In: *2011 IEEE International Conference on Communications (ICC)*. June 2011, pp. 1–5. DOI: 10.1109/icc.2011.5963095 (cit. on p. 33).
- [MW16] T. Mundt and P. Wickboldt. "Security in building automation systems - a first analysis." In: *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*. June 2016, pp. 1–8. DOI: 10.1109/CyberSecPODS.2016.7502336 (cit. on pp. 10, 58).
- [NG10] T. Novak and A. Gerstinger. "Safety- and Security-Critical Services in Building Automation and Control Systems." In: *IEEE Transactions on Industrial Electronics* 57.11 (Nov. 2010), pp. 3614–3621. ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2028364 (cit. on pp. 6, 33).
- [Nor05] Stephen Northcutt, ed. *Inside network perimeter security*. eng. 2. ed. XXXII, 734 S., Ill., graph. Darst., 23cm. Indianapolis, Ind.: Sams Publishing, 2005. ISBN: 0672327376 | 0-672-32737-6 (cit. on pp. 30–32, 42).
- [Pat12] Jignesh D. Patel. "Algorithms for deep packet inspection." PhD thesis. Michigan State University, 2012 (cit. on pp. 32, 33).
- [Sok17] Frank Sokollik. *KNX für die Gebäudesystemtechnik in Wohn- und Zweckbau*. ger. Ed. by Ralph Seela. 6., neu bearbeitete und erweiterte Auflage. 1 Online-Ressource (284 Seiten), Illustrationen, Diagramme. Offenbach: VDE Verlag GMBH, 2017. ISBN: 9783800740550 | 978-3-8007-4055-0 | 9783800740338 (print) | 978-3-8007-4033-8 (cit. on pp. 6, 8, 10, 11, 13–15, 17, 19, 20, 22, 24, 25, 29).
- [SP18] Wolfgang Schwab and Mathieu Poujol. *The State of Industrial Cybersecurity 2018*. Last accessed on March 14th 2019. June 2018. URL: <https://ics>.

BIBLIOGRAPHY

kaspersky.com/media/2018-Kaspersky-ICS-Whitepaper.pdf (cit. on p. 3).

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig verfasst. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht und ist in ähnlicher oder gleicher Weise noch nicht als Prüfungsleistung zur Anerkennung oder Bewertung vorgelegt worden.

Rostock, den 19. März 2019

Appendices

Appendix **A**

Disc with supplementary material