KNX IPv6: Design Issues and Proposed Architecture

Stefan Seifried, Günther Gridling, Wolfgang Kastner
Automation Systems Group
Technische Universität Wien
{sseifried, ggridling, k}@auto.tuwien.ac.at

***Abstract*—KNX is a worldwide open standard for home and building automation, which originated from a classical fieldbus protocol. While the needs for interacting with the upcoming Internet of Things (IoT) were already partially addressed by the recent *KNX Web Services* application note, KNX IP networking still supports IPv4 only.**

This paper proposes the usage of IPv6 networking as native KNX medium. The most obvious approach of mapping the existing *KNXnet/IP* mechanisms from IPv4 to IPv6 is neglected, and more elaborate solutions to existing issues in *KNXnet/IP* networking are discussed. The main objective is a better utilization of IPv6 features for a possible KNX IPv6 architecture, resulting in increased performance and compatibility to standard Information and Communication Technology (ICT) networking.

Therefore, several core design issues are raised and discussed in the course of this work. A transparent mapping approach between KNX and IPv6 addresses is proposed to facilitate the huge IPv6 address space. Further, the introduction of multi-homed Internet connectivity and node mobility by the usage of the Locator/ID Separator Protocol (LISP) is examined. Security considerations are addressed by comparing the upcoming *KNX IP Secure* standard to IPSec network layer security. Finally, the feasibility of the proposed approaches is shown with the help of a proof-of-concept implementation deployed in a small scale IPv6 testbed network.

I. Introduction

KNX is a worldwide communication standard for home and building services. Its key tasks are dedicated to support the automation of different trades, like heating, ventilation, air conditioning and lighting/shading. The KNX standard was formed by a merger of three competing European technologies, European Installation Bus (EIB), BatiBus and European Home Systems (EHS). Since then, KNX was recognized by the EN and ISO as official international norm. The core concept of KNX evolves around a bus-topology based field protocol to reduce cabling efforts in functional buildings. The emergence of Ethernet and IP as a common commodity encouraged the integration into control networks to further facilitate monitoring and engineering. Hence, the *KNXnet/IP* and the *KNX IP* sub-standards were created to enable information exchange between isolated fieldbus segments and further to integrate the IP protocol as an additional medium into KNX.

However, the integration efforts of KNX were solely targeted at the aging IPv4 protocol. While this design decision was feasible, given the prevalence of IPv4 over IPv6 at the time, IPv4 suffers from some critical limitations like insufficient stateless address configuration, size of the address pool and Internet protocol level security [1]. Solutions to the mentioned problems have been addressed by the IPv6 standard, and were backported to IPv4. However, the address space issue could not be resolved without introducing a breaking change into the IPv4 standard. Therefore, Network Address Translation (NAT) was introduced to IPv4 networks to allow for more usable address space. But even deep nested hierarchical NAT structures do not allow for a similar address amount as IPv6. Instead, the network and engineering complexity were increased. IPv4 NAT is also an intended (i.e. security) or unintended stopgap, which complicates access to services and communication objects behind a NAT gateway.

Since IPv6 has recently reached a two-figure adoption rate, it has become also of increasing importance for KNX to be IPv6 compatible. Furthermore, IPv6 is seen as one of the key enabling technologies for the upcoming Internet of Things (IoT) [2] and its properties allow for solutions that have been unthinkable with IPv4. For instance, IPv6 enables hassle-free multiple stakeholder scenarios like the smart grid. A smart meter deployed in a building needs to be accessible by different parties, like the utility company, building owner, tenants or a facility management system. So far, the NAT stopgap sealed off private network infrastructure and a bypass-solution resulted in considerable configuration effort. With IPv6 however, such a smart meter can have several IPv6 addresses, each with different security properties. The absence of NAT, which has been dismissed for IPv6 in RFC "IAB Thoughts on IPv6 Network Address Translation" [3], should not be considered as a security shortcoming. Instead, IPv6 offers IPSec as a built-in network layer security mechanism which allows for transparent end-to-end security.

This paper explores the design issues and discusses possible solutions for a KNX protocol extension resting upon the IPv6 standard. Section II provides a short recap of the state of the art in KNX IP development. Followed by Section III, where possible key aspects of the KNX protocol are revisited and their transferability to IPv6 concepts is motivated. Further, Section IV is concerned with the mapping of KNX addresses to IPv6 addresses and how the proposed Locator/ID Separation Protocol (LISP) protocol standard helps to adapt legacy protocols to IPv6. Security considerations for a KNX/IPv6 protocol are discussed in Section V, and how IPSec can increase the security even when there are existing measures already in place. Finally, an overview of the proof-of-concept implementation and the used testbed is provided in Section VI. The paper is concluded by an outlook for future work in Section VII.

II. KNX IP NETWORKING

In the last decades, the ever increasing usage of Ethernet facilitated the interconnection of diverse services and devices by the use of existing home and office networks. While Ethernet integration was traditionally achieved by deploying gateway solutions, the ongoing tendency is to promote end-to-end communication channels using IP-based protocols.

The trend in Ethernet integration also encouraged the creation of the KNXnet/IP standard, which was the first attempt to leverage existing Ethernet network infrastructure as a backbone network for KNX fieldbus segments. In general, the concept of KNXnet/IP is built around a *tunnelling protocol* concept where the KNX protocol frames are encapsulated inside IP frames. KNXnet/IP foresees two means of communication: On the one hand, *KNXnet/IP Tunnelling*, which facilitates acknowledged unicast transmission for reliable message exchange between a KNXnet/IP server and clients. On the other hand, *KNXnet/IP Routing*, that utilizes multicast transmission to mimic KNX fieldbus group communication mechanisms and distribute messages between communication nodes connected to the network. The purpose of *KNXnet/IP Tunneling* is situated in the engineering of KNX networks, whereas *KNXnet/IP Routing* is focused on runtime operation (i.e. exchange of process data).

As already mentioned, *KNXnet/IP Routing* is built upon IPv4 multicast mechanisms. Therefore a multicast address (224.0.23.12), part of the *AD-HOC Block I* has been registered at the Internet Assigned Numbers Authority (IANA). The corresponding *KNXnet/IP router* device which evolved from the *line coupler* is used at the KNX fieldbus to interconnect different fieldbus lines. It utilizes two network interfaces, one Twisted Pair (TP) fieldbus interface and one Ethernet network interface. Incoming messages from the TP interface are translated into IPv4 frames, that are sent using the IPv4 multicast mechanism to an arbitrary number of connected *KNXnet/IP routers*. In the other direction, KNX frames are extracted from telegrams received at the Ethernet interface and transmitted via the fieldbus network interface.

The KNX fieldbus has limited bandwidth compared to an Ethernet connection. Hence, it is beneficial to limit message forwarding to the fieldbus to prevent overload conditions. A *KNXnet/IP router* forwards messages to all devices in the same multicast group as per definition of the IP multicast mechanisms. Therefore, the *KNXnet/IP* standard foresees a *group filtering* mechanism, where telegrams can be rejected by their destination *KNX group address* at a *KNXnet/IP router* device.

The next step in the evolution KNX Ethernet connectivity was taken by the introduction of the *KNX IP* standard on top of the already established *KNXnet/IP* specification. This step was followed by the introduction of native KNX devices directly interconnected by the means of the *KNXnet/IP* transport mechanisms. Such a device was designed as a *KNXnet/IP router* directly attached to an application. This allowed the reuse of the existing *KNXnet/IP* network stacks, while directly forwarding the KNX frames to an application [4]. Hence, network management services could be left unmodified and easy integration of new *KNX IP devices* into existing *KNXnet/IP* infrastructure was enabled (see Figure 1).

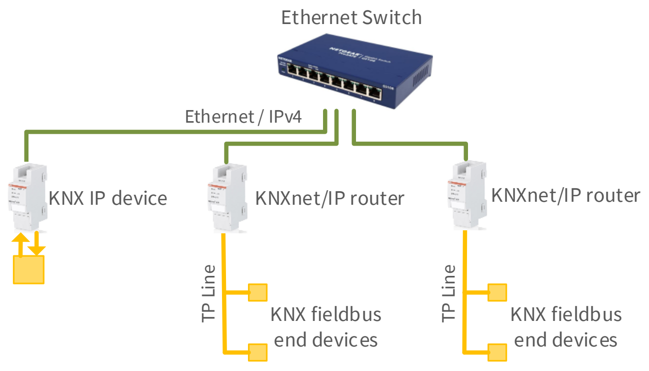

Fig. 1. KNX IP Topology

III. DESIGN ISSUES

The introduction of IPv6 as a foundation for KNX Ethernet traffic allows for novel solutions to problems inherent to the IPv4 nature of *KNX IP*. Since IPv6 was designed with IPv4 backwards compatibility in mind, the most direct approach to this problem is a simple port of *KNX IP* and *KNXnet/IP* on top of IPv6. However, such a solution would lay useful IPv6 features aside and would not result in any improvement to existing issues with *KNX IP*.

The foremost design issue of a KNX protocol based on IPv6 is how to make effective use of the huge address space. Whereas in IPv4 address space was scarce with only 254 devices per class C subnet, IPv6 allows to map the whole 16-bit KNX address space comfortably into the smallest IPv6 subnet range of 64-bit. Thus, it is feasible to automatically infer an IPv6 address from a given KNX address by following a predetermined mapping schema. The design issues and considerations for such a transparent mapping between KNX and IPv6 addresses is further discussed in Section IV.

An important subitem of the address mapping problem is dealing with KNX group communication. Devices, which are in a functional relation (e.g. light switching), exchange information by publishing data to a distinct group address. Such an address depicts a certain function in a distributed application, which is specified by the KNX standard in form of functional blocks (and communication objects). In *KNXnet/IP* and *KNX IP*, all group communication traffic had to be carried out via a single IPv4 multicast address. This again was due to the limited availability of IPv4 multicast addresses. A possible mapping schema is examined in Section IV-B.

The ongoing proliferation of the IoT and cloud computing led to an increasing demand in seamless connectivity between entities from Information and Communication Technology (ICT) and from the field of Building Automation Systems (BAS). But there has also been an increasing demand in quality of IP networking. In traditional KNX field networks, distributed applications were almost always contained within the same network segment. Hence, disruptions to the network only had limited effect, since the local functionality remained operable. The introduction of entities from ICT resulted in a greater demand of reliable Internet connectivity. Another concept introduced by ICT, that is foreign to BAS, are mobile devices. Until very recent, the network topology for KNX and BAS in general was considered static. Hence, there was neither the possibility for the seamless integration of new devices to an existing installation, nor the relocation of a device to a different location. The KNX/IPv6 architecture proposes a solution to the mentioned problems in form of the LISP protocol described in Section IV-C and Section IV-D. However, this will not allow the addition of new devices during runtime, since this cannot be done without introducing new management services to the KNX application layer. But it will allow for device mobility within the KNX/IPv6 network. On the other hand, it enables the utilization of multiple Internet access methods to increase reliability. Section IV-E describes the details of the suggested integration of *KNX/IPv6* into LISP.

Finally, information security is a predominant design issue for any network protocol. Security issues are usually classified by the CIA security targets, which are *confidentiality*, *integrity* and *availability* [5]. The *KNXnet/IP* protocol did neither prescribe any security measures nor had any noteworthy built-in security mechanisms included until very recently. However, the ongoing trend of connecting BAS networks to shared office networks or the Internet led to an increased security awareness. This resulted in the development of the *KNX Data Secure* and *KNX IP Secure* proposed standards, which are currently described in form of an application note [6]. *KNX Data Secure* defines a payload encryption schema across all supported KNX media, whereas *KNX IP Secure* is specifically designed to secure *KNXnet/IP* datagrams. However, *KNX IP Secure* seems to be redundant to existing security mechanisms of the IP protocol suite. Therefore, Section V aims at the examination of the *KNX IP Secure* protocol extensions in conjunction with IPSec security.

IV. Address Mapping

In order to transport KNX telegrams over an IPv6 network, as opposed to tunneling them, we need to map any KNX destination address to an IPv6 address. That IPv6 address will have to observe the IPv6 addressing architecture while embedding a segment that maps uniquely to the KNX address. In the following, we will investigate how such a mapping could work for both KNX individual and group addresses.

A. KNX Individual Addresses

The format of IPv6 addresses is specified in RFC 4291 “IP Version 6 Addressing Architecture” [7] for the different address types, e.g. multicast, link-local unicast, or global unicast. As we consider the option of connecting two KNX sites over the Internet a requirement, we will map KNX individual addresses to global unicast IPv6 addresses. The format of such addresses is described in the RFC under section 2.5. Apart from the specification of different prefixes to indicate different address types, the most important point is that the IPv6 global unicast address must be composed of a 64 bit subnet prefix and a 64 bit interface identifier (IID). The subnet prefix will be assigned by the Internet provider, and ensures global routability. The IID part is under local administrative control, and must be assigned to uniquely identify every node in the subnet. For that, subsection 2.5.1. and appendix A of the RFC are especially relevant, which specify how the interface identifier (IID) part of the address is constructed.

According to RFC 4291, the IID part of a global unicast address must be constructed in the modified EUI-64 format [8]. An EUI-64 is a globally unique identifier governed by the IEEE, which is composed of two parts: An organizationally unique identifier (OUI) assigned by the IEEE, and an extension identifier, which is controlled by the organization to which the OUI is assigned to. There are three different variants of the EUI-64 format, namely MA-L with a 24 bit OUI, MA-M with a 28 bit OUI, or MA-S with a 36 bit OUI. An important side effect of this format is that special semantics of two bits in the OUI are transferred over to the IID, namely the U/L bit (6th from the left) and the I/G bit (7th from the left) [9]. The U/L bit indicates the scope of the OUI as global when set to 0 and local when set to 1. The I/G bit distinguishes between individual (if set to 0) or group address (if set to 1). Therefore, when an IID is generated according to the EUI-64 format, it inherits those bits, which are renamed in RFC 4291 as u (the U/L bit) and g (the I/G bit). Due to the mapping of an EUI-48 (the MAC address) to an EUI-64 specified by the IEEE, these bits end up in the IID at 6th and 7th position from the left. Note that the RFC specifies that in an EUI-64 based IID, the u bit is to be inverted.

Those bit assignments are observed in a number of RFCs – some even reserve the entire octet in which the bits are located (e.g. RFC 6052 “IPv6 Addressing of IPv4-IPv6 Translators” [10]). However, as has been pointed out in RFC 7136 “Significance of IPv6 Interface Identifiers” [11], the requirement of using the EUI-64 format and in particular the significance of the u and g bits should reasonably be restricted only to IIDs that are actually derived from IEEE assigned MAC identifiers. For all other IIDs, the u and g bit should not be considered to have special meaning, and indeed the entire IID should be considered an opaque value. RFC 7136 specifically obsoletes the relevant statement from RFC 4291: “The use of the universal/local bit in the Modified EUI-64 format identifier is to allow development of future technology that can take advantage of interface identifiers with universal scope.” ([7], section 2.5.1). The reasoning in RFC 7136 is that such developments have not materialized so far, and that furthermore, there is evidence of cases where global MAC addresses are in fact reused, in particular in virtual machine environments. This would make the u bit unreliable in distinguishing global from local scope IIDs in such cases anyway.

Despite of the above, an argument can be made for observing the semantics of the u and g bits. As far as the u bit is concerned, it depends on the situation in the local network: If MAC-based – or indeed any EUI-48 or EUI-64 based – IIDs are in use, it might still make sense to retain the option of setting it to zero, corresponding to an U/L bit in the OUI/EUI-64 set to 1 to indicate local scope. While our KNX-mapped IIDs are indeed local, the intention here would be primarily not to indicate scope, but rather to provide separation from IEEE-assigned global IIDs, which will have the u bit set to 1. So, enforcing a cleared u bit could help in avoiding collisions particularly with commonly used MAC-derived IIDs.

As for the g bit, its intended semantics to distinguish individual and group addresses cannot directly be utilized in a KNX mapping. Although there is a distinction between individual and group addresses in the KNX address space as well, group addresses need to be handled differently in the IP routing system. This requires a different format, which will be described in more detail below. A possible reason to still keep the g bit it in the same place is the (admittedly speculative) possibility of transforming an IID back to a MAC-format link layer address to use the g bit in some form of traffic engineering, since switches are generally aware of multicast MAC addresses. In this paper, we will therefore assume that both the u and g bit are reserved in order to retain the option of using them if necessary.

As long as the formatting requirements are observed, mapping of the KNX address itself is rather straightforward. In order to keep the area/line/device hierarchy aggregatable, the KNX address fields are simply right-justified in the IID, and are ordered area, line, and device from left to right (Figure 2). The rest of the IID is free to be used for a local identifier, which needs to be unique within the site. This identifier is split into an upper part IDH and a lower part IDL by the u and g bits. At this point, there are no specific requirements on that ID part, other than local uniqueness. Still, it would probably make sense to reserve the IDH (the leftmost six bits) for flags, and to specify a type field in the IDL part to indicate that this is a KNX-based address, considering there are various other legacy address families which could be mapped to IIDs. Additionally, IDL should have at least one high-order bit set, so a KNX-based IID does not conflict with manually set IIDs which might be numbered upwards from 1, in line with the reasoning for inverting the u bit presented at the end of section 2.5.1 of RFC 4291.

Note that Internet-draft "IPv6 mapping to non-IP protocols" ([12], at the time of writing expired) proposes a mapping of legacy protocols to IPv6, which we initially used. It it is based on the IEEE mapping of EUI-48 to EUI-64 ([13]), but uses 0x0000 instead of 0xffff or 0xfffe as inserted value. However, in order to avoid conflicts with EUIs, the u bit should be cleared in any case, so restricting the format to use EUI-48 as a base does not seem to be necessary. On the other hand, if indeed global EUIs are to be used and the u flag is to be set, it would not be advisable to change the inserted value from the IEEE standard, as that could lead to collisions with EUI-64 IIDs.

B. KNX Group Addresses

In order to transmit KNX telegrams with group addresses as destination, IPv6 unicast cannot be used in a straightforward way. In a scenario where a KNX installation is composed of several KNX subnets connected over an IPv6 network, group telegrams need to be sent to all gateways. While this could be achieved by setting up a dedicated application layer protocol based on unicast messages, we are aiming at a direct mapping of KNX to IPv6 addressing, transparent to the IPv6 routing infrastructure.

An obvious candidate for KNX group addressing is therefore IPv6 multicast. A multicast address contains a group ID very much like the KNX group address. The basic multicast address format according to RFC 4291 specifies a 112 bit group ID field preceded by 16 format and flag bits. While a group ID of this size would allow for considerable flexibility, there is a major disadvantage to multicast addresses: They are not aggregatable, and therefore routing them over the Internet would put a considerable additional burden on already strained routing tables in the Default Free Zone (DFZ). Until this potentially critical scaling problem is solved, generic multicast addresses may be problematic.

Other RFCs revisit the multicast address format, e.g., RFC 3306 "Unicast-Prefix-based IPv6 Multicast Addresses", which extends RFC 4291 by partitioning the format further. The 112 bit group ID field is split up into 16 format and flag bits, a 64 bit unicast subnet prefix, and a 32 bit group ID field. The motivation here is to provide a way of assigning unique multicast group IDs without the need for an allocation protocol on a global scope. Strictly speaking, the 64 bit prefix and the 32 bit group ID together form the actual group ID, where the 32 bit ID is just the local part, and the – globally unique – prefix guarantees that the ID as a whole will be globally unique as long as the local part is unique within the subnet.

Note, however, that the scaling problem with routing multicast packets over the Internet still exists with RFC 3306. The embedded unicast prefix merely indicates in which subnet and thus administrative domain a multicast address was created. RFC 3306 specifies it to guarantee that the addresses are unique, but does not restrict their use to hosts within the assigning subnet. This means that such addresses are still not aggregatable and therefore require individual routing table entries.

Until a large-scale multicast implementation is established in practice, in theory a workable compromise might be to assume that an RFC 3306 based multicast address is actually bound to be used within its assigning subnet. This would make the prefix in the extended group ID usable for routing, and crucially eliminate the need for additional routing table entries, since the route would be the same as that for unicast packets with that prefix. However, routers would have to be able to

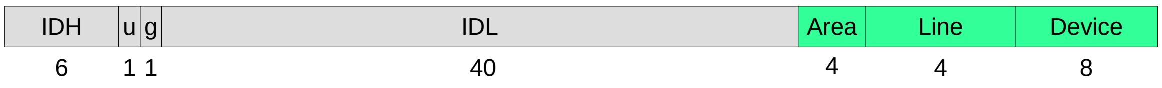

Fig. 2. KNX-based Interface Identifier

match the unicast prefix within a multicast address, where it is in a different position compared to unicast addresses. Of course, this means the address would not actually be multicast at the inter-site level. Then again, for management and security reasons, each site in a distributed KNX installation would need to be aware of its partner sites anyway, so replicating group telegrams individually would not require too much additional effort, and it might be worth to address the scaling issue.

Without any changes in inter-domain routing, the only scalable way would be to implement multicast between KNX sites by replicated unicast messages. The downside of this would be that the inter-domain routing system is now completely unaware that the packet is actually meant to be multicast, whereas in the mixed scenario above, the multicast address format and flags as specified in RFC 4291 and its updates would be visible to the inter-domain routing infrastructure.

A replicated multicast would require a dedicated KNX multicast router (KMR) within any site that contains KNX subnets. It would need to store the list of all KNX mapped group IDs as well as a list of the unicast prefixes for all KNX partner sites.

Multicast packets received from the local interfaces and addressed to one of the known KNX mapped group IDs would have to be replicated to all partner sites. For that, the destination address unicast prefix would have to be replaced with the prefix of the destination before sending the message individually to each partner site. If we assume that RFC 3306 based multicast with location restriction is not used on the inter-site level, then the multicast packed would be encapsulated within a unicast message; in this case, the KMR would need to store the individual addresses of each partner site's KMR instead of just the site prefix.

If a multicast packet is received from a partner site, the local KMR would receive it and could participate in reliability and/or security protocols, but no KNX-specific processing would be needed, since the target address is an RFC 3306 multicast address with the local unicast prefix. If an encapsulated multicast message is received, it would have to strip the outer header, replace the unicast prefix with the local prefix, and then send it to the local site.

For the mapping of KNX group addresses to IPv6 multicast groups, there are two basic approaches:

- Direct Mapping: In this case, KNX group addresses are mapped one-to-one to IPv6 group IDs. The group ID of an RFC 3306 multicast address is 32 bits wide, so the 16 bit wide KNX group addresses can easily be accommodated. The benefit of this would be that apart from using replicated multicast on the inter-site level, no KNX-specific protocol would be needed – and this change would only be necessary due to the scaling problem present in the current multicast architecture. A possible downside would be that both the KMR as well as all KNX gateways would have to join all or most of the IPv6 multicast groups.
- Set Mapping: To mitigate the need for the KMR and the gateways to join a possibly large number of multicast groups, and to reduce the number of multicast groups the routing system has to deal with, their number could be reduced by partitioning the KNX group address space and mapping entire subsets to IPv6 group IDs. To transmit an IPv6-mapped KNX group message, the KMR as well as each gateway would encapsulate it in a new header with the multicast group ID to which the original group ID belongs. Upon receiving such a message, the header would be stripped, and the inner header would be processed like a direct mapped group address. The benefit of this would be that fewer IPv6 group IDs are used, down to a minimum of one. However, each gateway would need to implement the additional header processing.

C. The Naming-Addressing Problem

A major drawback of current IP addresses is the fact that they overload naming and addressing in one common scheme. During early development of the Internet, this used to be a benefit: An IP address specifies where a particular device is located, and since addresses have to be unique anyway, they could conveniently double as device identifiers. That the name of a device was tied to its location seemed to be an acceptable compromise in the interest of efficiency, since throughout the network stack, no mapping of names to addresses was necessary.

However, with the increasing relevance of features like mobility and multi-homing, this has become a serious problem. Various work-arounds have been in place for some time now, see for example RFC 4116 "IPv4 Multihoming Practices and Limitations" [14]. Unfortunately, work-arounds like deployment of provider independent (PI) addresses for multi-homing do not scale well.

The decoupling of naming and addressing (often referred to as identifier/locator split) will be an important issue in the ongoing attempts to make sure inter-domain routing scales with changing requirements.

Essentially, the key idea is to use different concepts on different layers of the network stack, each suited to the requirements of that layer. Applications should use DNS names to identify hosts, since they are human readable, and changes are – relatively – cheap. The network layer should use identifiers, as having a namespace of its own would allow e.g. to encode semantic information about the device. The transport layer should use locators, which are optimized for routing. Of course, the necessary mapping of these different concepts in between the layers introduces some overhead; however, the fact that identifiers and locators can now be optimized specifically for their primary purpose should easily make up for that.

There are ongoing attempts to introduce an architectural separation of addresses from names/identifiers, e.g. RFC 6740 “Identifier-Locator Network Protocol (ILNP) Architectural Description” [15] or RFC 6830 “The Locator-ID Separation Protocol (LISP)” [16]. Both these architectures introduce a distinction between addresses and names, which in this context are referred to as ‘locators’ and ‘identifiers’.

It is not clear yet how the name/address overload problem will be solved, or whether any of the current proposals will eventually be adopted as the standard solution. It can, however, be expected that there will be a identifier/locator split in some form. At a minimum, this means that devices will be identified by their name, and that there must be a way to find the current locator for any given name.

In our context, this separation is relevant in so far as it allows us to use KNX addresses as names/identifiers. The only requirement is for us to have some control over the identifier assignment. Once the destination of a transmission can be referred to by the identifier instead of address, and a subset of the identifier space can be mapped to the KNX address space, KNX telegrams can be routed over the internet with very little additional effort required.

In the following, we will briefly describe how KNX telegrams could be handled in an environment with identifier/locator split, using a LISP network as an example.

D. The Locator-ID Separation Protocol (LISP) Overview

The Locator-ID Separation Protocol is designed to provide a partial identifier/locator split with minimal impact on existing network infrastructure, which makes it a good candidate for a first evaluation of possible KNX mapping mechanisms. Of course, it is not at all clear whether LISP or a similar protocol will ever be added to the Internet architecture – in the context of this paper, it is merely a convenient way to set up an identifier/locator split. LISP still uses IP addresses – or rather, their syntax – as identifiers, but separates the identifier space from the locator space. It also does not require any changes in the transport layer, using encapsulation and tunneling mechanisms to transport identifier-addressed packets.

Essentially, a LISP network consists of a set of individual LISP sites, which are connected through the Internet.

Within a LISP site, the identifier address space is used for routing, while between sites, the locator address space is used.

LISP identifiers are referred to as Endpoint IDs (EIDs). In the architecture as proposed by RFC 6830, they are still used a locators, but this use is restricted to within a LISP site. Therefore, they are specified to be in either IPv4 or IPv6 format, although in theory other addressing formats could also be used.

Locators in LISP are called Routing Locators (RLOCs), where routing refers specifically to inter-site routing. In order to transport a packet with an EID as the target address, the EID is mapped to an RLOC, and the packet is then encapsulated with a new header carrying the RLOC as the target address and tunneled through the IP network. This separation between EIDs and RLOCs obviously does not provide a full identifier/locator split. The intention is to make sure addresses used in the inter-site part of the IP network – and in particular the DFZ – are aggregatable, while allowing more freedom within LISP sites.

The mapping between EIDs and RLOCs is provided by LISP-specific devices called Map Resolvers (MRs) and Map Servers (MSs). An MR maintains a database mapping EIDs to RLOCs. An MR is essentially the client-side interface of an MS, split out into separate devices to keep requests more local and using caching for load distribution. Note that the mapping is not necessarily one-to-one: LISP specifies that a particular EID can be mapped to several RLOCs, with different priorities/weights for each of them.

To implement the tunneling, special routers are needed. In LISP, they are called Ingress Tunnel Router (iTR) and Egress Tunnel Router (eTR), where ingress and egress refer to the direction of a particular message transmission from one LISP site to another. Often, ingress and egress functionality will be integrated in the same device, then referred to as xTR. A message generated by a host within a LISP site is routed based on the EID until it reaches a border router to the Internet, which in this case functions as the iTR. The iTR checks its mapping cache and if necessary requests the RLOC for the given EID from an MR and then wraps the packet with an outer header using that RLOC as target address. The RLOC is the assigned IP address of a particular eTR, which is known to the MR/MS since each eTR publishes its EID to RLOC mappings. The packet is then routed through the inter-site network until it reaches the eTR, which strips the outer header and forwards the packet within its site using the EID as locator.

E. Integration of KNX and LISP

As mentioned above, the identifier/locator split approach opens up new ways to integrate legacy protocols with the Internet by allowing more control over the identifiers. With LISP in particular, we now have more freedom to map legacy address families to EIDs, since global routability is only relevant to RLOCs, but not EIDs.

As far as the mapping of KNX addresses to EIDs is concerned, the considerations in Section IV apply, as in our setting EIDs would be IPv6 addresses.

1) KNX Group Addressing in LISP: Of special note is the implementation of KNX group addressing as multicast. LISP according to RFC 6830 can make use of multicast group addresses, but does not map them in the same way as unicast addresses. Multicast addresses are used as both EIDs and RLOCs, that is, the outer header uses the multicast target address given in the original header, and only the source address is changed to the RLOC of the iTR. This

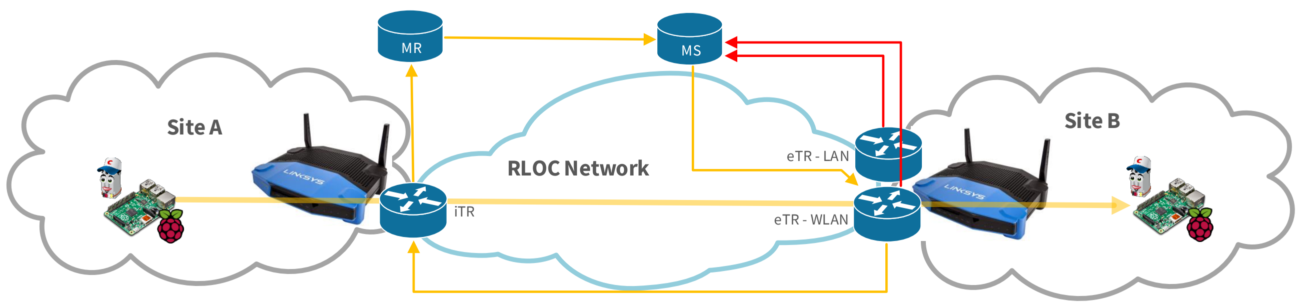

Fig. 3. LISP Site to Site

is because multicast addresses are independent of network topology, which means identifier/locator separation does not apply to them.

However, given that one EID can easily be mapped to many RLOCs and outgoing interfaces (OIFs) for traffic engineering purposes by the LISP mapping system, it would of course be possible to use that to implement multicast on an underlying unicast network by replicating the packet to multiple destinations, not just multiple OIFs. Some LISP implementations already provide such a feature, e.g. Cisco IOS with what is referred to as head-end replication. As mentioned above, this could be used as a workaround to implement scalable multicast between KNX sites. Basically, the LISP infrastructure would provide most of the functionality required for mapping of KNX group addresses and for the replicated multicast required on the inter-site level.

2) KNX in a LISP Environment: In order to integrate KNX networks with an existing LISP network, obviously a KNX Gateway is needed for each KNX network. The range of KNX addresses a particular gateway is covering would be specified with an IPv6-KNX mapping mask/ID, similar to the network mask/address of a CIDR address. That way, each gateway could be assigned a power-of-two subset of the KNX network, starting from area IDs through line IDs down to device IDs.

Each of these Gateways must be assigned an IPv6-KNX mapping mask big enough to cover all KNX device addresses that are connected to this Gateway. Note that ‘assigned’ in this case simply means that the addresses covered by this mask must not be in use elsewhere within the site.

In the simplest case, a Gateway assigns to its network interface an IPv6 address constructed from its IPv6-KNX mapping mask/ID and the device’s KNX address for each device connected to it. Each of these addresses becomes an EID, which is announced to the MS by the iTR. This is the most direct mapping of KNX addresses to the IP routing system. A major benefit of this would be that no changes would be required either at local routers, iTRs, eTRs, or inter-site routers. However, if a large KNX network is connected to a single Gateway, that Gateway would need to assign possibly thousands of IPv6 addresses to its network interface. While this is certainly feasible, it could be worked around by using e.g. ip6tables on the Gateway, and setting the appropriate routes at local routers.

If a Gateway receives a telegram with an individual address from the KNX network, it checks whether the address is covered by its IPv6-KNX mapping mask/ID. If so, then the address is internal, and need not be propagated to the IP network side. If the address is outside the mapping mask, the IPv6 address is formed by merging the mapping ID with the KNX address and then sent to the IP side. There, it becomes an EID, which is either routed locally to the destination, or is picked up by the iTR, which takes care of sending it to the appropriate remote eTR.

If a telegram to a group address is received, it is sent out to the IP side with the corresponding multicast address as destination. Locally, since it is a multicast address, all gateways which are registered for this group will receive it. It would then be the task of the iTR to distribute the packet to all the eTRs specified in the mapping system, either as multicast packets with target-specific unicast prefix, or encapsulated in unicast packets addressed to the individual eTRs. This mapping mechanism for multicast packets implements some of the functionality described above for a KMR, and it would be the main architectural difference to a standard LISP system. This would also be one of the main benefits of LISP in our setting – the mapping between identifiers and locators that is already in place at the iTRs allows control over the egress traffic, which facilitates KNX specific adaptations. In addition, the separate identifier space allows all KNX gateways to be under the same identifier prefix; it essentially puts them inside a virtual common domain, even though each site may have a different locator prefix.

V. KNXnet/IP Secure vs. IPv6/IPSec

The KNX protocol did not offer any security measures until very recently and solely relied on network isolation for the protection of its devices and data. While this seemed like a sufficient solution for KNX installations that only comprised of fieldbus network segments, the advent of *KNXnet/IP* effectively voided the network isolation principle. The installers of KNX networks were only advised to secure *KNXnet/IP* connections by either using separate physical Ethernet cabling or use other means to create a virtual private network between *KNXnet/IP* routers. Unfortunately, this was rarely done, so the *KNX Data Secure* and *KNX IP Secure* standards [6] were proposed to allow for security mechanisms within the KNX engineering process.

As mentioned in Section III, *KNX Data Secure* provides payload encryption on all physical layers included in the KNX standard. Whereas *KNX IP Secure* introduces a new security wrapper service, that encapsulates *KNXnet/IP* frames in the payload of an IP datagram. This work mainly focuses on the *KNX IP Secure* standard, since it is especially designed to secure the *KNXnet/IP* protocol and competes with IPv6 network layer security to some extent.

A. KNX IP secure

KNX IP Secure specifies an Advanced Encryption Standard (AES) 128-Bit encryption mechanism with counter mode and Cipher Block Chaining Message Authentication Code (CBC-MAC) is utilized as an encryption algorithm for both unicast communication (*KNXnet/IP* Tunneling) and multicast communication (*KNXnet/IP* Routing). In terms of key management, an Elliptic Curve Diffie Hellman (ECDH) key exchange algorithm with two pre-shared keys is used for unicast communication (KNXnet/IP Tunneling). However, for multicast communication, only a single pre-shared key is used.

There are several critical issues with the design of the KNX security protocols, most noteable the unlimited lifespan of the keys used to encrypt multicast communication. Therefore, a single compromised *KNXnet/IP* device allows the decryption all group communication within a single KNX network. Further description of possible attack vectors and weaknesses in the KNX security standards can be found in [17].

In conclusion, the *KNX IP Secure* standards are certainly a step in the right direction. However, the suitability to secure traffic in possible hostile networks (i.e. the Internet) can be doubted, and security measures should be complemented.

B. IPSec

IPSec, on the other hand, was built into IPv6 from the very beginning and was even mandatory at one point in time [18]. The main advantage of IPSec is its transparency to any application layer protocol, since it is strictly contained in the network layer of the ISO/OSI model. This allows a conflict free side-by-side coexistence of KNX security standards and IPSec. However, there have also been problems with IPSec. The rather convoluted standard documents led to varying implementations by different vendors and hence to interoperability problems. Luckily, IPSec has matured and the described problems are now a thing of the past.

The IPSec security architecture consists of the following components:

- **Authentication Headers (AH)** [19] provides mechanisms for datagram integrity and authenticity, but no payload encryption.
- **Encapsulating Security Payload (ESP)** [20] implements features to encrypt, authenticate, and check data integrity. It also offers a anti-replay service by utilizing message sequence numbers.
- **Security Associations (SA)** [21] supply necessary configuration information for both AH and ESP within the scope of the Internet Security Association and Key Management Protocol (ISAKMP) framework.

Further, IPSec offers two modes of operation, namely *tunnel mode* and *transport mode*. The *tunnel mode* creates a new packet, which encapsulates the original packet in an encrypted form. *Transport mode* on the other hand, injects the IPSec header into the original header and encrypts the payload. The benefit of *tunnel mode* is a complete encryption of the transported packet, whereas basic address information remains readable using *transport mode*. However, *tunnel mode* also requires more resources and possible adds more latency to network communication than *transport mode*.

C. KNX/IPv6 security approaches

A prime target of the KNX/IPv6 architecture is the support of a one-to-one mapping between KNX fieldbus group communication and IPv6 multicast groups. While IPSec already mentioned in RFC 4301 "Security Architecture for the Internet Protocol" [18] the support for multicast datagram SAs in *transport mode*, RFC 5374 "Multicast Extensions to the Security Architecture for the Internet Protocol" [22] also defines support for *tunnel mode* and group key management protocols. While both *tunnel mode* and *transport mode* are viable options for use in a KNX/IPv6 security architecture, a resource constrained nature of embedded KNX/IPv6 devices must be assumed. Hence, *transport mode* offers a better trade-off between security and resource usage.

In conclusion, there are three possible scenarios for a KNX/IPv6 architecture:

- *KNX IP Secure* only
- *KNX IP Secure* in conjunction with IPSec
- IPSec only

A "*KNX IP Secure only*" solution is the most direct approach in a transition movement from *KNXnet/IP*. But as previously outlined, it does not improve on the current security situation of the KNX protocol and inherits the same flaws as as mentioned in [17].

The "IPSec only" approach offers a non-intrusive approach to KNX IP security at the network layer. However, IPSec configuration needs to be an intrinsic part of KNX engineering and hence additional configuration services need to be made available. Legacy compatibility through gateways would be much more cumbersome, since frames secured by *KNX IP Secure* standard have to decrypted first.

Finally, the "*KNX IP Secure* in conjunction with IPSec" offers the best compatibility between the existing KNX IPv4 approaches and possible IPv6 based security. However, the double encryption of network packets might prove to be too resource intensive.

VI. Proof of Concept

The practical feasibility of the presented approaches was evaluated by implementing a proof-of-concept KNX IPv6

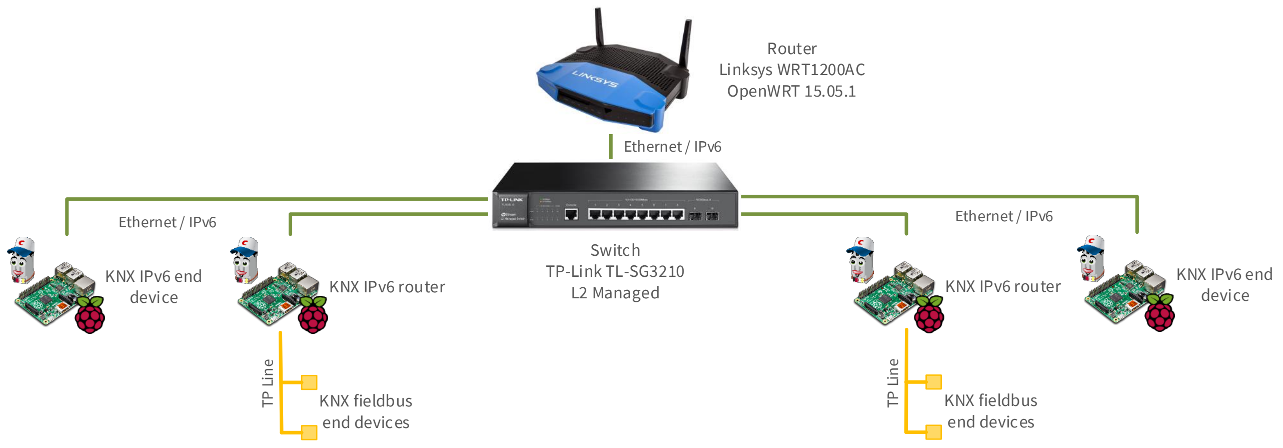

Fig. 4. Proof-of-Concept Topology

network stack. Therefore, an existing KNXnet/IP router open-source implementation, namely *Calimero*[1], was extended with the solutions and IPv6 features as discussed by this work. Further, the KNX IPv6 network stack was deployed as a *KNX IPv6 end device* and as a *KNX IPv6 router* to a *Raspberry Pi* single board computer. Connectivity between the *Raspberry Pi* and KNX field bus line was established by using a TPUART USB dongle.

A detailed description of the testbed hardware and software is worthwhile, since some parts of the utilized IPv6 feature set is either marked experimental by the IETF or not widely implemented by network gear manufacturers. However, only a rather small number of devices were incorporated in the testing environment. Hence, collecting network performance data would not yield meaningful results and therefore network service quality measures are deliberately not presented. An overview of an exemplary network segment is shown in Figure 4. The central elements of the network infrastructure are the IPv6 enabled layer 2 switch, *TP-Link TL-SG3210* and the *Linksys WRT1200AC* router running OpenWRT.

Despite IPv4/IPv6 dual stack capable, the *TP-Link TL-SG3210* was selected for its Multicast Listener Discovery (MLD) snooping capabilities ([23], [24]). This allows the routing device to examine MLD packets and forward those telegrams based on their content. The MLD snooping technique enables significant reduction of IPv6 multicast traffic by routing multicast packets only to LAN ports that are subscribed to the concerned multicast group.

Due to the experimental status of the LISP protocol there are only a few networking devices available which provide at least a basic implementation. While enterprise products by CISCO offer the most complete support for LISP including a *Map Server* and *Map Resolver*, routers running the OpenWRT[2] firmware are a cost-effective and versatile alternative. The core of the OpenWRT LISP implementation is provided by the Open Overlay Router (OOR) project [3]. The utillized LISP infrastructure for feasibility tests was provided by the LISP beta network[4].

[1]https://github.com/calimero-project
[2]https://openwrt.org/

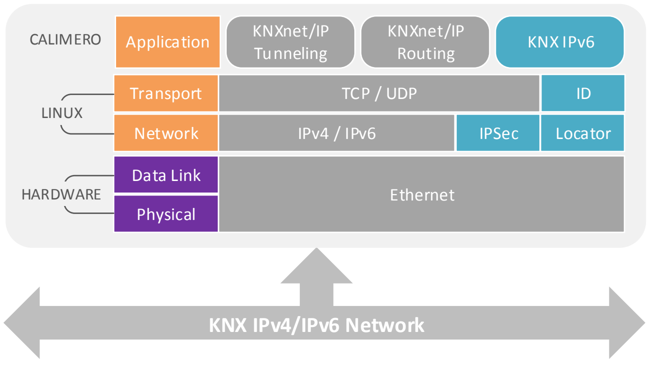

Fig. 5. KNX IPv6 Network Stack

Figure 5 depicts a component diagram of the utilized network stack, which has been deployed to a Raspberry Pi hardware platform to create both KNX IPv6 router and KNX IPv6 end devices. The IPSec and LISP network and transport layer functionality is accomplished by using readily available software components provided by the *Linux* environment. On the one hand, IPSec security is achieved by using the *Strongswan*[5] application. The *Locator/ID* functionality, on the other hand, is simply achieved by configuring the mapped IPv6 addresses via the KNX IPv6 application and *Linux* network interface management mechanisms.

VII. Conclusion and Outlook

The paper presents an overview of design issues and evaluation of architectural choices for an upcoming KNX IP standard based on the IPv6 protocol. While a direct transmission of

[3]https://github.com/OpenOverlayRouter/oor
[4]http://www.lisp4.net/beta-network/
[5]https://www.strongswan.org/

the existing KNXnet/IP and KNX IP standards is possible, it would make inadequate use of the IPv6 feature set.

Despite fixing shortcomings in the existing KNX communication standards, the declared goal was a broad utilization of approved or proposed IPv6 standards. On the one hand, this facilitates better compatibility in scenarios where the IP medium is shared with other services. On the other hand, IT administrators are enabled to govern KNX IPv6 networks with little or even without knowledge of KNX networking due to the use of well-known IPv6 management mechanisms.

The prime design issues for a KNX/IPv6 architecture are address translation between field networks, group communication, reliable Internet transport and security. Existing approaches and concepts of the KNX protocols have been revisited, and possible mappings to IPv6 techniques have been shown. As for the address translation problem, a transparent mapping of KNX addresses to IPv6 addresses is possible due to the huge IPv6 address space. Further, the KNX group communication concept can be directly mapped to IPv6 multicast communication for the very same reason. Reliable Internet transportation is addressed via LISP, which offers a versatile approach to enable multi-homing. Finally, security is discussed in form of a comparison of the upcoming *KNX secure* standards with IPSec.

The suggested approaches have been evaluated for their feasibility with a proof-of-concept KNX/IPv6 network stack in conjunction with a testbed network. While the focus was on the proof of viability of the discussed features, the scalability of the design choices remains to be shown. Hence, a next step in the pursuit of KNX/IPv6 would be a large-scale test of the proposed concepts in conjunction with a diverse set of network equipment from different vendors.

Furthermore, IPv6 enables new features for the KNX standard, that might prove useful in upcoming IoT use-cases. Besides multi-homing, LISP was designed to enable device mobility. Hence, the proposed KNX/IPv6 architecture also enables location independent KNX devices. However, performance and design issues regarding such mobile KNX devices is not part of this work and remains to be shown.

References

[1] A. N. A. Ali, "Comparison study between IPV4 and IPV6," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 314–317, May 2012.

[2] S. Ziegler, C. Crettaz, L. Ladid, S. Krco, B. Pokric, A. F. Skarmeta, A. Jara, W. Kastner, and M. Jung, *IoT6 – Moving to an IPv6-Based Future IoT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 161–172.

[3] D. Thaler, L. Zhang, and G. Lebovitz, "IAB Thoughts on IPv6 Network Address Translation," RFC 5902 (Informational), Internet Engineering Task Force, Jul. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5902.txt

[4] G. Neugschwandtner and W. Kastner, "Congestion control in building automation networks: Considerations for KNX," in *2009 35th Annual Conference of IEEE Industrial Electronics*, Nov 2009, pp. 4149–4154.

[5] R. Shirey, "Internet Security Glossary, Version 2," RFC 4949 (Informational), Internet Engineering Task Force, Aug. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4949

[6] KNX Association, "KNXnet/IP Secure - Application Note 159/13 v04," September 2013.

[7] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291 (Informational), Internet Engineering Task Force, Feb. 2006. [Online]. Available: https://tools.ietf.org/rfc/rfc4291.txt

[8] IEEE Standards Association, "Guidelines for 64-bit Global Identifier (EUI-64)," IEEE Standards Association. [Online]. Available: http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

[9] ——, "802 IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture," IEEE Standards Association, Mar. 2002.

[10] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li, "IPv6 Addressing of IPv4-IPv6 Translators," RFC 6052, Internet Engineering Task Force, Oct. 2010. [Online]. Available: https://tools.ietf.org/html/rfc6052.txt

[11] B. Carpenter and S. Jiang, "Significance of IPv6 Interface Identifiers," RFC 7136, Internet Engineering Task Force, Feb. 2014. [Online]. Available: https://tools.ietf.org/html/rfc7136.txt

[12] G. Rizzo, A. Jara, A. Olivieri, Y. Bocchi, M. Palattella, L. Ladid, S. Ziegler, and C. Crettaz, "IPv6 mapping to non-IP protocols," Internet-Draft (Informational), Internet Engineering Task Force, Mar. 2015. [Online]. Available: https://tools.ietf.org/id/draft-rizzo-6lo-6legacy-03.txt

[13] IEEE Standards Association, "Guidelines for 48-bit Global Identifier (EUI-48)," IEEE Standards Association. [Online]. Available: http://standards.ieee.org/regauth/oui/tutorials/EUI48.html

[14] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill, "IPv4 Multihoming Practices and Limitations," RFC 4116 (Informational), Internet Engineering Task Force, Jul. 2005. [Online]. Available: https://tools.ietf.org/rfc/rfc4116.txt

[15] R. Atkinson and S. Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description," RFC 6740 (Experimental), Internet Engineering Task Force, Nov. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6740.txt

[16] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The Locator/ID Separation Protocol (LISP)," RFC 6830 (Experimental), Internet Engineering Task Force, Jan. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6830.txt

[17] A. Judmayer, L. Krammer, and W. Kastner, "On the security of security extensions for IP-based KNX networks," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, May 2014, pp. 1–10.

[18] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4301

[19] S. Kent, "IP Authentication Header," RFC 4302 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4302

[20] ——, "IP Encapsulating Security Payload (ESP)," RFC 4303 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4303

[21] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296 (Internet Standard), Internet Engineering Task Force, Oct. 2014. [Online]. Available: https://tools.ietf.org/html/rfc7296

[22] B. Weis, G. Gross, and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol," RFC 5374 (Proposed Standard), Internet Engineering Task Force, Nov. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5374

[23] R. Vida and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," RFC 3810 (Proposed Standard), Internet Engineering Task Force, Jun. 2004. [Online]. Available: https://tools.ietf.org/html/rfc3810

[24] H. Holbrook, B. Cain, and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast," RFC 4604 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4604