The IEEE International Conference on
Industrial Informatics (INDIN 2008)
DCC, Daejeon, Korea July 13-16, 2008

Gateway-free Integration of BACnet and KNX using Multi-Protocol Devices

Wolfgang Granzer, Wolfgang Kastner, Christian Reinisch
Vienna University of Technology, Automation Systems Group
Treitlstrasse 1-3, Vienna, Austria
Email: {w,k,cr}@auto.tuwien.ac.at

***Abstract*— Today, building automation systems can be realized using a multitude of different standards. Since each of these standards has its benefits, the combination of different standards and consequently of their best features promises substantial synergies. However, an integration is far from being straightforward and thus demands research. Starting from an analysis of the benefits that integrated building automation networks can offer, this work reviews the different approaches to combine KNX and BACnet. A gateway-free solution based on multi-protocol devices is considered most promising and examined in detail. The required protocol adaptations are discussed and, finally, a prototype implementation of an integrated, gateway-free BACnet/KNX internetwork is presented.**

I. Introduction

Building Automation Systems (BAS) aim at improving control and management of mechanical and electrical systems in buildings. The system functionality is broken up into three levels [1]. At the field level, the data is collected (measurement, counting, metering) and the process is controlled (switching, setting, positioning). The automation level encompasses the various aspects of automatic control, e.g., the execution of control loops. Global configuration and management tasks (e.g., visualization) are part of the management level functions.

Today, many different standards for BAS exist. The three most popular and well-known ones are KNX [2], LonWorks [3] and BACnet [4]. While these open standards are application-independent and can be used at all three levels, other standards are dedicated to the use at a single level. Some of the latter ones are application-specific (e.g., the Digital Addressable Lighting Interface, DALI [5]) while others bring along specific characteristics (e.g., ZigBee focuses on wireless networking [6]). A mapping of the most important standards to the architectural levels in BAS is illustrated in Fig. 1. In Europe, it can be observed that both KNX and LonWorks are used predominantly at the field and automation level while BACnet is prevalent at the management and automation level. However, the introduction of KNXnet/IP and LonWorks/IP made way for these protocols' use also at the management level. Likewise, the specification of MS-TP and LonTalk as network option enabled BACnet to be used at the field level. Especially, in North America, BACnet/MS-TP is used at the field level.

Nowadays, the three level functional model [7] is often implemented as a flatter, two level architecture. On the one hand this shift stems from the development of intelligent field

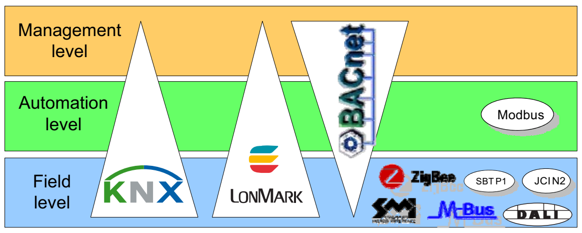

Fig. 1. Standards in building automation

devices which incorporate more functionality and thus can take over automation functions themselves. On the other hand information technology is no longer used exclusively at the management level but also at the automation level. Therefore, the functions of the automation level can be split and reassigned to the "intelligent" field as well as the management devices. Today's building automation networks (BANs) are often implemented following this two tier model: multiple field networks that are home for intelligent sensors, actuators and controllers are interconnected by a common backbone. A typical example of such a 2-tiered BAN is shown in Fig. 2.

II. BACnet and KNX

The main objective of BACnet is to provide a solution for building automation and control systems of all sizes and types. BACnet was first published in 1995 by the standing standard project committee SSPC135 of the American Society of Heating, Refrigerating and Air ConditioningEngineers (ASHRAE). Together with ANSI, BACnet was made an ISO standard in 2003. The currently effective standards are ANSI/ASHRAE 135:2004 [4] and ISO 16484-5:2007 [8].

The protocol architecture of BACnet consists of four layers which correspond to the physical, data link, network, and application layer of the ISO/OSI model. In contrast to other building automation standards, BACnet is not bound to a specific network medium. Hence, BACnet leaves the underlying physical and data link layer undefined, so that – in principle – any physical/data link layer combination could be used. However, to increase the compatibility of BACnet devices offered by different vendors, six physical/data link layer combinations called *network options* have been defined by BACnet. These are *Ethernet, ARCNET, Master-*

Slave/Token-Passing (MS/TP), LonTalk, Point-to-Point (PTP) and *BACnet over IP (BACnet/IP)*.

In 2002, the KNX standard was defined as a combination of EIB (European Installation Bus), Batibus and EHS (European Home System). In 2004, KNX was published as an European Standard (EN 50090). Finally, in 2006, it became an international standard (ISO/IEC 14543). The standard itself is maintained by the Konnex Association [2].

KNX provides the choice of different network media. It supports the use of twisted-pair (KNX TP0 and KNX TP1), powerline (KNX PL110 and KNX PL132) as well as a wireless solution called KNX Radio Frequency (KNX RF). Additionally, a simple form of IP tunneling is also available (KNXnet/IP). The most common type is KNX TP1 which allows a maximum bandwidth of 9.6 KBit/s and free topology. In KNX, two different types of communication are possible. On the one hand, configuration and maintenance of devices is accomplished using unicast connections (connection-less or connection-oriented). On the other hand, process data is exchanged among communication groups using exclusively multicast communication (group communication).

Up to now, a KNX based network option is not part of the BACnet standard. However, KNX offers specific advantages over other protocols. Amongst others, these are free topology, simple yet robust medium access using CSMA/CA and a small network stack footprint which makes KNX an interesting option especially at the field level. The promising combination of the features of BACnet and KNX in a so called BACnet internetwork is shown in Fig. 2. The KNX segment is interconnected to the BACnet backbone using a gateway. Following the 2-tier approach, the use of a gateway is mandatory as it caters for the mapping of KNX and BACnet data points. But the use of gateways also exhibits several drawbacks. Most prominent, the configuration and maintenance effort increases as all relevant data points from both adjacent network segments have to be translated. This requires considerably large mapping tables to be stored and may be a limiting factor regarding the scalability of the BAS. The gateway approach thus introduces a single point of failure and additionally a security risk. Regarding security, all process data has to be concentrated in the gateway device to be accessible from both attached networks. This makes the gateway of particular interest for an adversary who gains a complete system view once the gateway is attacked successfully.

Faced with these drawbacks, a solution that allows the interconnection of a KNX network with a BACnet backbone without the need for gateways is desirable. This approach and its advantages are presented in the next section. In Sections IV and V the mechanisms that enable our solution are presented. The discussion is followed by a prototype implementation in Section VI. An outlook on future work is given in Section VII.

III. Gateway-free integration

Since a gateway-based approach is responsible for several disadvantages, a gateway-free solution is targeted. Still, the functionality originally provided by the gateways has to remain available in the network to allow communication across BACnet and KNX network segments.

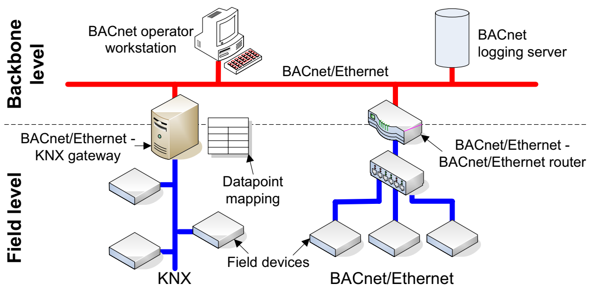

Fig. 2. Two tier model using a gateway

One way to integrate KNX would be to use BACnet communication over the KNX medium for the field networks. The drawback of this solution is the obvious need for completely new devices. All of them would have to feature the KNX physical and data link layer below the mandatory higher BACnet layers. The KNX part would thus be degraded to a BACnet network option and standard-conform KNX communication up to the user application would no longer be possible. This clearly contrasts the goal to fully integrate a KNX network that can coexist with the BACnet network. Still, this approach would allow to eliminate the gateways.

The only way to realize a coexistent and integrated BACnet/KNX network without gateways is to break down the data point mapping to the field level. This distribution of the gateway functionality across the devices only requires a modification to the devices that shall be integrated. These devices have to implement both a BACnet and a KNX protocol stack to fully support BACnet as well as standard KNX communication. The result is a so called *multi-protocol device* that supports both communication protocols [9]. Such a BACnet/KNX multi-protocol device can coexist with standard KNX devices attached to the same network segment without any modification and can even fully participate in the KNX communication. Following this approach the data point mapping previously performed by gateways located at the network segment borders becomes obsolete. This makes way for the substitution of the gateways with more simple routers. These routers analyze and translate the incoming network messages only up to OSI layer 3 and therefore need not be aware of any data points. Compared to gateways, configuration and maintenance of routers implies considerably less effort as no mapping tables need to be maintained. The reduced complexity of a “routing task” also impacts the hardware requirements (especially memory and computational power) which are significantly decreased for the dedicated routing device. Clearly, the configuration effort at the field level is in turn slightly increased. Field devices are now themselves required to keep a mapping table containing all data points to be shared. However, these data points and especially their associated bindings have to be configured and maintained in any case so that only the data point mapping table has to be considered additional overhead. This overhead is alleviated by the fact that the tables can be created and maintained at least semi-automatically with the help of management tools. Furthermore, breaking down the data point mapping to the field level also reduces a potential security risk that is introduced by the use of gateways. Compared to a gateway, a multi-protocol device has to store only a subset of the data points of particular interest. Therefore, an adversary who has successfully attacked a multi-protocol device only gains access to a subset of the process data.

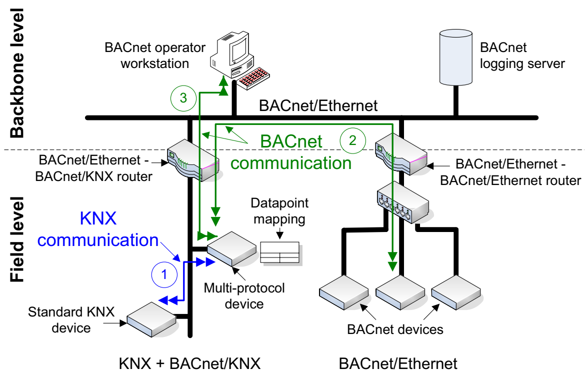

Fig. 3. Two tier BACnet internetwork with integrated KNX network

Fig. 3 shows an example of a BAN where a KNX field network is integrated with a BACnet network using routers exclusively. The router based solution allows standard KNX devices as well as BACnet/KNX multi-protocol devices to share the same KNX field network.

Four different types of data exchange in the integrated network are possible. Obviously, multi-protocol devices are able to exchange data with other multi-protocol devices. Additionally, a multi-protocol device that is attached to a KNX network can exchange process data with other standard KNX devices (cf. ①), because it implements a full KNX protocol stack. As each multi-protocol device also implements a BACnet stack, full support for all BACnet communication services is given. This main feature allows two additional ways of data exchange. As shown in ②, the multi-protocol device is able to exchange process data with other BACnet field devices located in other BACnet field networks. The necessary mapping is then performed by the multi-protocol device itself and the messages just need to be routed to the destination segment. Finally, also BACnet management devices (e.g., a BACnet operator work station located at the backbone) can directly access the multi-protocol device to perform configuration or maintenance tasks (cf. ③).

The router-based approach offers significant advantages over the integration with the help of a gateway. Nevertheless, some precautions that prevent unwanted interference between BACnet and KNX network messages have to be taken. These, and a detailed description of the multi-protocol device architecture are presented in the following section.

IV. BACnet/KNX multi-protocol device

The key components of gateway-free integration are multi-protocol devices. Their main benefit is the ability to communicate with standard KNX devices as well as with BACnet devices that are possibly located in other BACnet networks. To achieve this, multi-protocol devices implement both communication protocol stacks.

Fig. 4 shows the structure of a *BACnet/KNX multi-protocol device* named *BACknx* device. Once integration of a KNX network is intended, BACknx devices need to have an interface to the KNX network medium. Thus, the use of KNX as transport medium for both communication stacks is mandatory. Due to the fact that BACnet does not specify the data link and physical layers, KNX must be used as network option for BACnet. A detailed description will be given in Section V.

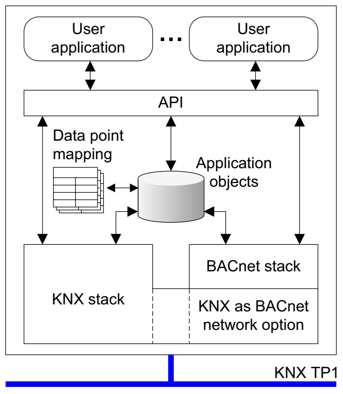

Fig. 4. BACnet/KNX multi-protocol device

On top of the two communication stacks, a common data base that stores the application data is located. The following application model is assumed in this approach: process data (e.g., sensor and actuator values) are represented by data points. Each device has a set of data points that are represented by application objects within the device. In order to achieve compatibility between the application models of BACnet and KNX, the following application object model is used: each application object consists of two mandatory properties (object ID and object type), various object-specific properties (e.g., present sensor value) and multiple optional properties (e.g., min. value, max. value). Each property can be either a single data element or an array of elements. In addition, each property has a property ID and a field specifying the property type i.e., the size of a single data element.

Obviously, the object and property IDs and types as well as the encoding of the stored data values differs in BACnet and KNX. Therefore, a mapping and/or translation scheme needs to be defined. This is achieved through the employment of four different tables for mapping the object IDs (including the application service access points), the object types, the property IDs, and the property types. An example mapping of a binary application object is presented in Section VI-A.

V. KNX as network option for BACnet

Because KNX has not yet been considered as network option for BACnet, an approach to use the popular KNX medium TP1 is part of our proposed solution. The BACnet network layer provides an unconfirmed communication service which in turn requires (at least) an unacknowledged communication service from an underlying data link layer. Regarding the use of the KNX data link and physical layers as new BACnet network option this means that the KNX `L_Data` service [2] can be used. The corresponding parameters of the `L_Data` service are the source and destination address, a message priority, the user data (i.e., the LSDU) to be transmitted, and a Boolean value specifying if a layer 2 acknowledgment is expected. Setting up the required point-to-point communication within a local BACnet network communication can be accomplished using KNX's individual addressing scheme. In this approach, the KNX individual address can be used unchanged as BACnet device address.

Both BACnet and KNX distinguish between four priority levels. However, the levels have different semantics so that a mapping has to be performed. With respect to the fact that BACnet high priority messages have to be handled first, the following mapping is chosen: `BACnet normal` to `KNX low`, `BACnet urgent` to `KNX normal`, `BACnet critical equipment` to `KNX urgent`, and `BACnet life safety` to `KNX system`. The remaining BACnet user data are embedded into the LSDU part of the `L_Data` service. Although the use of KNX link layer acknowledgments for BACnet/KNX frames is not mandatory, its activation is recommended to increase robustness.

This straight-forward mapping of BACnet NPDUs into the KNX LPDUs works as long as exclusively BACnet communication is used. However, in case that BACnet and KNX communication shall be used simultaneously, problems would emerge. Consider, for example, that a BACnet broadcast message is transmitted over the KNX network. Since both BACnet-over-KNX and standard KNX messages use the same LPDU type, it is not possible to differentiate between a "mapped" BACnet-over-KNX and a standard KNX message. Therefore, standard KNX devices receive the BACnet broadcast message, too. However, the message will be interpreted wrongly by the higher KNX layers.

One way to counter this problem would be to define a new LPDU type for BACnet/KNX messages. The drawback of this solution is that standard KNX routers (couplers) could no longer route these messages since the new LPDU type is unknown to them. To overcome this limitation, a new TPDU type called `T_DATA_BACNET_REQ_PDU` has been defined. Using this new TPDU type, an unwanted interference between BACnet/KNX and KNX messages is prevented and the message can still be processed correctly by standard KNX routers. The encapsulation (replacement of the TSDU and mapping of parameters) is handled by a so called *BACnet/KNX Virtual Link Layer* (BKVLL) which is located between the KNX transport and the BACnet network layer (cf. Fig. 5).

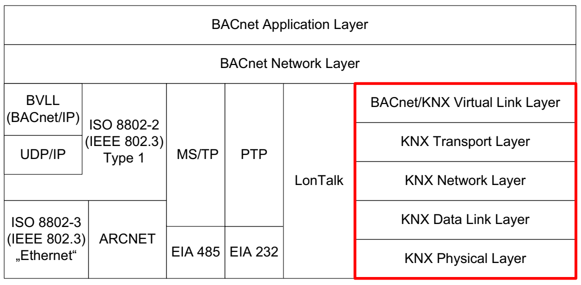

Fig. 5. BACnet/KNX as new network option

BACnet supports message segmentation at the application layer. The maximum amount of APDU segments may vary and can be negotiated between the sending and receiving device. Due to the fact that different network options can be used in BACnet, the maximum length of a single APDU within a single message segment may vary, too. According to the BACnet specification, the maximum APDU length is limited by the maximum APDU length transmittable by the device, the maximum APDU length accepted by the remote peer and the maximum NPDU length permitted by the local, remote or any intervening network segments between the sending and the receiving device. In addition to these constraints, BACnet demands a maximum APDU length of at least 50 octets. Since the APDU length of standard KNX frames is limited to 15 octets, they cannot be used without further precautions.

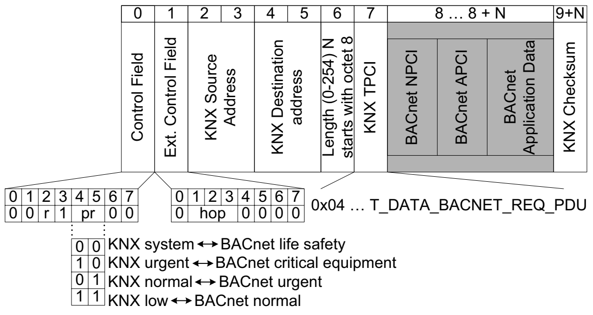
0 1 2 3 4 5 6 7 8 … 8 + N 9+N
Control Field
Ext. Control Field
KNX Source Address
KNX Destination address
Length (0-254) N starts with octet 8
KNX TPCI
BACnet NPCI
BACnet APCI
BACnet Application Data
KNX Checksum
0 1 2 3 4 5 6 7
0 0 r 1 pr 0 0
0 1 2 3 4 5 6 7
0 hop 0 0 0 0
0x04 … T_DATA_BACNET_REQ_PDU
0 0 KNX system ↔ BACnet life safety
1 0 KNX urgent ↔ BACnet critical equipment
0 1 KNX normal ↔ BACnet urgent
1 1 KNX low ↔ BACnet normal

Fig. 6. BACnet/KNX frame format

One solution would be to introduce segmentation at the data link layer that is performed independently of the segmentation at the BACnet application layer. This additional segmentation process would have to be implemented by the BKVLL. The main drawback of this solution is the resulting overhead. Since the maximum NPCI length of a BACnet NPDU (in case of a transmission between two BACnet/Ethernet peers) is 24 octets, the resulting NPDU length is 74 (assuming a maximum APDU length of 50). Together with 1 octet occupied by the segmentation sequence number, this leads to a minimum length of 75 octets. Therefore, the BACnet NPDU would have to be split into five KNX standard messages since a single standard KNX TSDU may only contain up to 15 octets. The encountered overhead of this approach is unacceptably high at 55%.[1] However, the overhead introduced by the data link layer segmentation can be avoided if the KNX extended frame format is used. Using KNX extended frames, the KNX TSDU can now contain up to 254 octets. Considering a worst case NPCI length of 24 octets, 230 octets remain available for the APDU. This way, data link layer segmentation is clearly rendered unnecessary. Fig. 6 shows the frame structure of the proposed solution.

VI. Prototype implementation

As a first proof-of-concept, a prototype BACnet internetwork consisting of a BACnet/Ethernet backbone and a KNX TP 1 based field network has been set up (cf. Fig. 7).

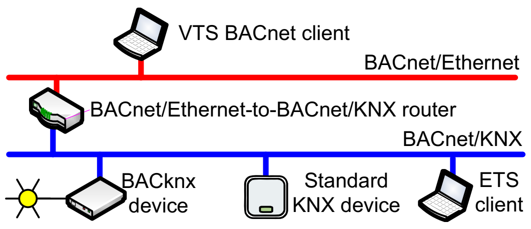

Fig. 7. Prototype implementation

The encapsulation technique presented in Section V enables (separate) BACnet as well as standard KNX communication without any interference between the two protocols. However, also mixed operation of both protocols is supported. For demonstration purposes additional devices are integrated into the prototype network. At the BACnet/Ethernet network, a BACnet client is connected via Ethernet. The client runs the Visual Test Shell (VTS)[2], an open-source Windows application for BACnet conformance testing. The VTS is used to send and receive BACnet messages.

The KNX TP1 network contains a standard KNX light switch and a KNX management client (represented by a notebook) running the standard KNX management software Engineering Tool Software (ETS). Furthermore, an implementation of the previously presented *BACknx* device is attached to the KNX TP1 network. Obviously, a router that interconnects the BACnet/Ethernet backbone with the KNX TP1 network is required. Its implementation is presented in Section VI-B.

A. BACknx device

The main hardware components of the BACknx device are a MSP430 microcontroller and a Twisted Pair - Universal Asynchronous Receive Transmit (TP-UART) [10] interface which is used to access the KNX TP1 medium. Additionally, multiple digital I/O ports including connected peripherals (LEDs and buttons) are available. In the proposed test environment, one LED is used to simulate a light source.

[1]In this case, the overhead is calculated as follows: $overhead = (KNXLPCIlength + KNXNPCIlength + KNXTPCIlength + sequenceID)/(BACnetNPDUlength) * 100$.

[2]http://sourceforge.net/projects/vts/

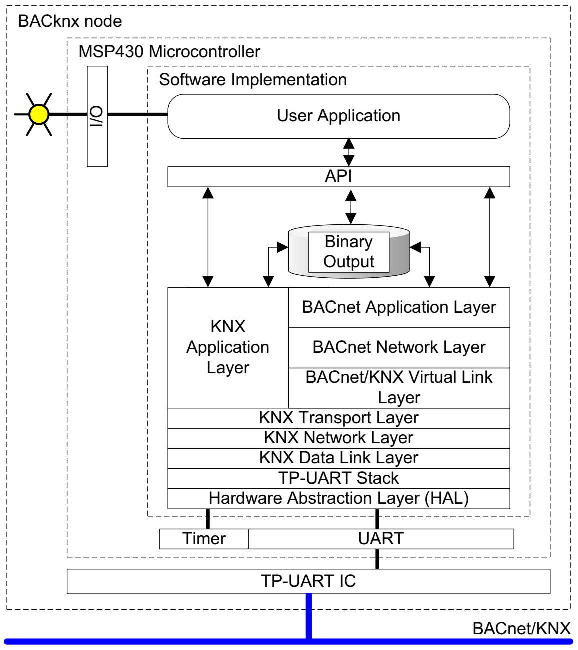

Fig. 8. BACknx node

The software architecture of the BACknx device is shown in Fig. 8. A simple *Hardware Abstraction Layer (HAL)* abstracts the underlying microcontroller hardware (i.e., UART and timer). On top of it, a hardware independent *TP-UART* stack allows an efficient implementation of the BACnet/KNX protocol. In principle, the TP-UART supports the use of extended frames. However, due to the format of the TP-UART specific services, a maximum frame length of 62 octets and thus a KNX APDU length of 53 octets is possible [10]. Therefore, a data link layer segmentation is still necessary. The maximum BACnet APDU length of 50 octets (as it is required by the BACnet specification) can be guaranteed by splitting a BACnet NPDU into at most two KNX messages, thus keeping the overhead significantly lower.

The *BACknx protocol stack* resting above consists of two major parts: A lean *KNX stack* provides basic services for group communication. In particular the services `A_GroupValue_Read`, `A_GroupValue_Write` and `A_GroupValue_Response` as well as the connection-oriented services `A_PropertyValue_Read`, `A_Property Value_Write` and `A_PropertyValue_Response` are implemented. Furthermore, a simple *BACnet stack* supports the BACnet services `Who-is`, `I-am`, `Who-has`, `I-have`, `ReadProperty`, and `WriteProperty`. The BACnet stack is placed on top of the BACnet/KNX virtual link layer which is responsible for replacing the TSDU part of a KNX message with the BACnet NPDUs.

On top of the KNX and BACnet stack, a simple, generic application object table has been implemented. For this prototype the table contains a single binary output object that is associated with the state of a LED. The object can be accessed in three different ways: On the one hand, the user application is associated with the binary output object. Whenever the state of the binary output object is changed (i.e., by an incoming BACnet or KNX request), the user application is notified thereof and can change the state of the LED. On the other hand, the object is accessible through both communication stacks using BACnet or KNX services. In the prototype, this means that either the VTS client can alter it using a BACnet `PropertyWrite`, or the KNX light switch can be used to turn on/off the LED by sending a KNX `A_GroupValue_Write`. Additionally, the current state of the object can be read out by the VTS (using BACnet's `ReadProperty`) or by the ETS (via KNX's `A_GroupValue_Read`).

As mentioned in Section IV, a data point mapping has to be performed. In the current implementation, the KNX binary group object is mapped to a BACnet binary output object. The group address of the KNX group object is mapped to the BACnet `Object_Identifier` while the value is mapped to the BACnet object property `Present_Value`. The mapping of the property types follows the data point type mapping described in Annex H.5 of the BACnet specification [4].

The presented prototype implementation requires about 35Kbyte of memory. Compared to a KNX-only implementation that uses about 30Kbyte, the BACnet stack needs an additional 5Kbyte of memory, and therefore introduces a minimum overhead only.

B. *BACnet/Ethernet-to-BACnet/KNX router*

For successful operation of an integrated BACnet/KNX internetwork, a router that interconnects the network segments is needed. The task of the BACnet/Ethernet-to-BACnet/KNX router is to receive messages from either network, convert the messages according to the destination network's requirements and finally forward the converted messages to the destination network. As shown in Fig. 9, the router has been implemented on an ARCOM VIPER embedded PC. A connection to the BACnet/Ethernet network is established using the onboard Ethernet controller. The KNX TP1 is accessed through a TP-UART board that is connected to an EIA 232 interface.

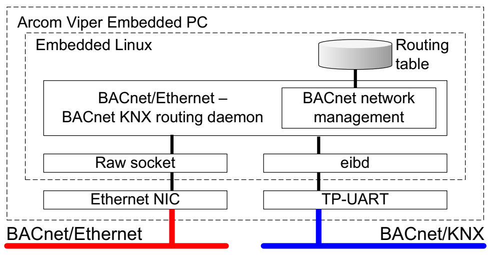

Fig. 9. BACnet/Ethernet-to-BACnet/KNX router

Embedded Linux has been chosen as operating system for the router device. A *routing daemon* routes the network traffic based on the information about the network topology that is stored in the routing table. Currently, this table only contains one entry for the Ethernet and one for the KNX network. However, an extension for routers that support more than two network interfaces is easily possible. The main task of the routing daemon is to receive, convert and forward incoming BACnet/KNX or BACnet/Ethernet messages. Incoming KNX frames are captured using *eibd*[3] whereas BACnet frames are received using a local *UNIX raw socket*. Additionally, the daemon supports multiple BACnet network layer communication services. Currently, services for a discovery of the router by other BACnet devices (i.e., `Who-Is-Router-To-Network` and `I-Am-Router-To-Network`) as well as for dynamically changing the routing table (i.e., `Initialize-Routing-Table` and `Initialize-Routing-Table-Ack`) are implemented.

VII. Conclusion and Future Work

Integration is one of the main future challenges in BAS, yet it offers significant benefits. This paper presents an approach to integrate KNX field networks with BACnet. This concerns, on the one hand, KNX TP1 as new BACnet network option and, on the other hand, multi-protocol devices.

Further development is, thus, possible in two ways. First, adapting the encapsulation mechanisms to add support for the other KNX media (e.g., KNX RF) is easily possible. Second, the modular architecture of the multi-protocol devices allows a straightforward extension towards other BAS standards, e.g., wireless networks such as ZigBee.

Furthermore, the presented mapping between BACnet and KNX application objects will be refined. Starting from the basic mapping defined in Annex H of the current BACnet specification, also the mapping of more complex BACnet object types to KNX functional blocks (and vice versa) shall be included in our generic application model. Finally, the BACnet conformance testing tool VTS shall be extended to support KNX as new network option.

Acknowledgment

The work presented in this paper was funded by FWF (Austrian Science Foundation) under the project P19673.

References

[1] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.
[2] "KNX specification," Version 1.1, 2004.
[3] "Control network protocol specification," ANSI/EIA/CEA 709.1, 1999.
[4] "BACnet – a data communication protocol for building automation and control networks," ANSI/ASHRAE 135, 2004.
[5] "A.C.-supplied electronic ballasts for tubular fluorescent lamps – Control interface for "Control by digital signals"," IEC 60929 Annex E, 2006.
[6] *ZigBee Specification 2007*, ZigBee Alliance, San Ramon, 2007.
[7] International Organization for Standardization, "Building Automation and Control Systems (BACS) – Part 2: Hardware," ISO 16484-2, 2004.
[8] "Building automation and control systems (BACS) – part 5: Data communication protocol," ISO 16484-5, 2007.
[9] S. Soucek and D. Loy, "Vertical Integration in Building Automation Systems," in *Proc. 5th IEEE INDIN*, June 2007, pp. 81–86.
[10] Siemens, "Technical Data EIB-TP-UART-IC," 2001, version D.

[3]http://www.auto.tuwien.ac.at/~mkoegler/index.php/eibd