58

KNX: A Worldwide Standard Protocol for Home and Building Automation: State of the Art and Perspectives

Michele Ruta
Politecnico di Bari

Floriano Scioscia
Politecnico di Bari

Giuseppe Loseto
Politecnico di Bari

Eugenio Di Sciascio
Politecnico di Bari

58.1 Introduction

The home and building automation (HBA) is a technological effort aiming to make houses and buildings more controllable, autonomous, and comfortable. Recently, the demand for occupant comfort, flexibility in the management of a building equipment, and the need for efficient energy use have become pressing issues. Safety and comfort requirements, along with energy management, can be achieved only if an intelligent control and monitoring of devices and appliances is performed. A wide array of building automation and control systems (BACS) have been developed and implemented in commercial products, focused on improving energy consumption and minimizing waste and maintenance (reflecting guidelines of the European Standard EN 15232 [1]).

An HBA system requires a large deployment of sensors and actuators in order to detect contextual information and then transfer control data to all building components. A point-to-point interconnecting building device is impractical as it largely increases the amount of wires and consequently the installation complexity and the related costs. A solution is in a deployment of a system using a common, and shared, communication medium by exploiting a well-known and dependable protocol.

BACS are based on bus infrastructures identified as the most effective technological solution for interconnection and communication. Over time, a plethora of protocols have been progressively developed and deployed. Nowadays, the ISO/IEC 14543-3 EIB/KNX [2] (KNX in short) can be recognized as a de facto standard for HBA. It joins and enhances three legacy bus standards, namely, EIB, EHS, and BatiBUS. A KNX medium (twisted pair [TP], radio frequency [RF], power line [PL], or IP/Ethernet) can interconnect heterogeneous bus devices making them able to share and exchange information in a uniform way. Each KNX component can be seen as a sensor detecting an environmental situation (weather stations, lighting sensors, monitoring systems, presence sensors) or an actuator controlling building equipment such as blinds or shutters, safety appliances, energy managers, heating, ventilation and air-conditioning systems, and multimedia devices. Every appliance can be monitored, supervised, and signaled via a shared protocol without the need for single extra control centers.

58.2 KNX: Short History

In 1997, three main European associations were involved in the HBA centered around on-a-bus communication among devices and appliances. Batibus Club International (BCI) was a French no-profit association promoting the BatiBUS medium originally developed by Schneider Electric. European Home Systems Association (EHSA) was a Dutch association promoting the EHS technology, resulting from a European project aiming at the automatic configuration of bus-compatible white (washing machine, cooker) and brown goods (video, hi-fi). European Installation Bus Association (EIBA) was a Belgian cooperative society developing the EIB technology implemented by a consortium of manufacturers headed by Siemens. EIBA consisted of shareholders (*members*) and users of the technology (*licensees*). EIBA developed and marketed a configuration tool called EIB Tool Software.

The fusion of the mentioned associations resulted in a new association called *KNX Association*, and its final aim was the definition of a common standard for a new bus system named *KNX* [2]. It had the following basic features:

- A backward compatibility with EIB
- A simple and effective configuration method essentially based on a PC procedure exploiting ETS but also using central controllers, code wheels, and so on
- The possibility to include heating, ventilation, and air-conditioning (HVAC) systems among supported appliances
- The support for RF and IP media

The KNX standard was approved by the European Committee for Electrotechnical Standardization (CENELEC). In April 1999, the KNX Association was officially founded in Brussels by nine members. Currently, it groups over 340 members in 37 countries (March 2014), including companies that were previously not a member in any of the promoting associations.

In May 2002, the first version of the KNX specification was released among the KNX members starting the KNX Certification Scheme for Products. The intellectual property rights (IPR) clearance was effective from June 2003. As a result, all KNX members may use the KNX technology free of any patent claims of fellow members in KNX-certified products. In 2003, the national committees of CENELEC Technical Committee TC205 (*Home and Building Electronic Systems*) approved a subset of the KNX specification referred to as KNX media and KNX stack, making it a European Standard (as part of the EN 50090 bus family standard). In 2006, the same documentation was approved again by the European CEN for application in BACS. It is registered as EN 13321-1, series EN 50090. The KNX idea to tunnel and route KNX telegrams across IP networks (see later on for details), that is, the KNXnet/IP protocol, was also approved in 2006 by CEN as EN 13321-2, and finally, in the same year, the EN 50090 series achieved the international recognition as ISO/IEC 14543-3. In 2007, the Chinese translation of KNX specification reached the status of national standard as GB/T 20965.

Today, the KNX Association is responsible for all the activities related to the KNX system, and in particular, it oversees the following:

- The specification work conducted by the expert groups
- The technical hotline support to members
- The international standardization of novel features and characteristics but also the KNX certification by incoming organizations
- The logo protection
- The management of the training documentation
- The advertising and communication strategy (e.g., publication of the official association journal twice a year, diffusion of brochures, organization of biannual award for best KNX projects, presence at fairs)
- The management of national groups, that is, groups of member companies in a given country responsible for the local promotion of the KNX system
- The scientific partnership, that is, the involvement of universities, research centers, and technical institutes in scientific and educational initiatives as the annual organization of the KNX Scientific Conference
- The management of the partnership for trained and certified contractors

58.3 KNX System Specification

The KNX specification covers all technical elements of the standard, ranging from the low-level device details and certification rules to testing and application descriptions. In what follows, the fundamentals of the protocol will be outlined according to the KNX specification, version 2.0 [2]. Supported transmission media will be briefly surveyed and both meaning and function of protocol layers defined in the standard will be clarified. Then a description of the reference network architecture and the addressing features will be reported, followed by a section related to the integration of the KNX protocol with IP networks. Finally, details about application-level objects will be given.

58.3.1 Transmission Media

The KNX system offers the possibility for the manufacturers to choose between several transmission media, according to market requirements and specific habits. Moreover, it is also possible to combine them to build multimedia and multivendor network configurations.

The basic TP and PL communication media are accompanied by the RF support in order to make KNX networks flexible and adaptable to multiple application domains and installation situations. Some of the main features of supported media are presented in the following:

- TP TP1 is the basic medium inherited from the EIB protocol. It provides a solution for wired cabling, using a safety extralow-voltage (SELV) network and a supply system. KNX supports both TP1-64 and TP1-256 media, which differ in the number of connectable devices per physical segment (64 or 256). TP1-256 is backward compatible toward TPI-64. The *TP1* transmission rate is 9600 bit/s with a character-oriented asynchronous data transfer mode and a half-duplex bidirectional communication. It also complies with the carrier sense multiple access with collision avoidance (CSMA/CA) protocol. All network topologies (e.g., line or star) may be used and mixed.
- PL PL110 is also inherited from the EIB protocol and enables communication over the main power supply network of a building. It is based on an asynchronous transmission of data packets and a half-duplex bidirectional communication. PL110 uses the central frequency of 110 kHz with a 1200 bit/s data rate; it complies with CSMA and the EN 50065-1 standard [3].
- RF enables a communication via radio in the 868.3 MHz band for short-range devices. It supports the frequency-shift keying, a maximum duty cycle of 1%, and a Manchester data encoding.

KNX allows integrated solutions for IP-enabled media like Ethernet IEEE 802.2 [4], Wireless LAN IEEE 802.11 [5], and FireWire IEEE 1394 [6] exploiting the KNXnet/IP protocol as explained in Section 58.3.5.

The KNX *v*1.1 specification also supported TP0 and PL132 media to ensure a fully backward compatibility with EHS and BatiBUS protocols. Those media are no longer available in the new KNX *v*2.0 protocol. The related documentation was phased out in the last specification and both TP0 and PL132 cannot be used in applications for the KNX certification.

58.3.2 Protocol Layers

The KNX protocol defines a seven-layer stack compliant with the ISO/OSI model [7]. On top of the physical layer, a common kernel model is shared by all devices of the KNX network. The reference structure is shown in Figure 58.1.

The *physical layer* is on top of the communication channel. It mainly consists of a logical unit, a medium attachment unit (MAU), and a medium interface (i.e., the connector). Each medium needs a dedicated MAU and a suitable logical unit. The MAU is used for encoding the logic signals to physical ones and vice versa. In the transmission mode, the logical unit serializes each data octet in a bit sequence; then, it frames the obtained data and transforms them in an asynchronous timed logical signal. In the receiving mode, the logical unit reverts the obtained signal in a data bit stream to be checked, rebuilt, and aggregated into several data octets. Finally, the medium-independent sublayer of the logical unit transfers the received octets to the upper layer.

A general *data link layer*, built above each data link layer specific for each medium, provides a high-level medium access and logical link control. It also takes responsibility for the transmission of single KNX frames between two or more devices on the same subnetwork.

During transmission, the data link layer (i) builds a complete frame starting from data coming from the network layer, (ii) gains access to the medium according to the particular access policy in use, and (iii) transmits the frame to the data link layer of each peer entity, using the services of the physical layer.

When receiving, the data link layer (i) detects possible corruptions in the frame, (ii) transfers the frame to the upper layers, and (iii) issues positive or negative acknowledgments back to the transmitting data link layer entity.

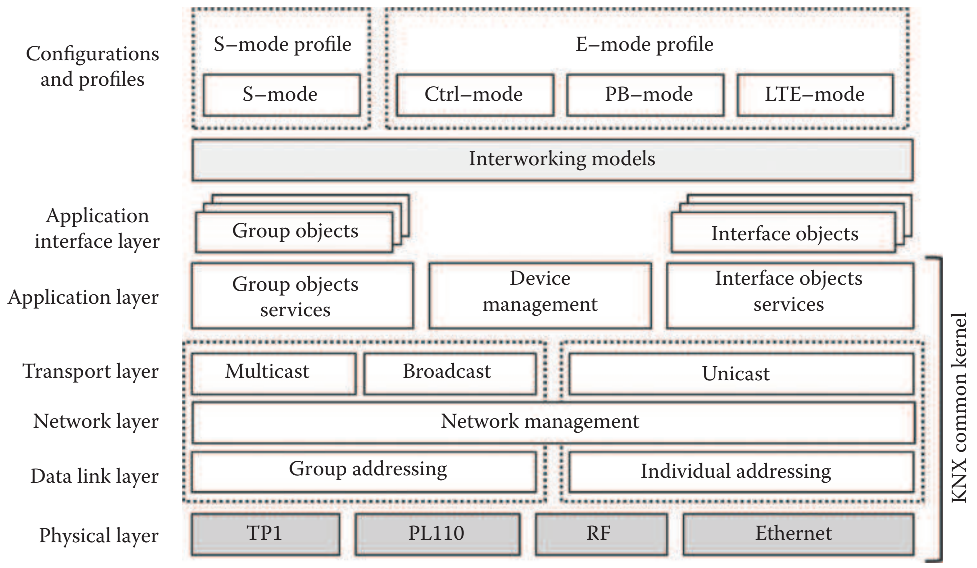


FIGURE 58.1 KNX protocol stack and reference architecture.

The data link layer also manages unicast, multicast, and broadcast communication options and device addressing, as described in Section 58.3.3.

The *network layer* takes care of managing communications across KNX subnetworks. Hence, its role is relevant mainly to the nodes with routing functionalities. As an example, this layer provides a segment-wise acknowledged telegram and controls the hop count of a frame.

The *transport layer* enables the data transmission over different communication modes. Five kinds of communication are defined in the standard:

1. Point-to-multipoint, connectionless (multicast)
2. Point-to-domain, connectionless (broadcast)
3. Point-to-all-points, connectionless (system broadcast)
4. Point-to-point, connectionless (unicast)
5. Point-to-point, connection oriented (unicast)

Finally, *session* and *presentation layers* are unused, while the *application layer* offers several services to application processes. Each different service depends on the communication used at the transport layer. Usually services related to point-to-point communication and broadcast mainly refer to network applications, whereas services related to multicast are used for runtime operations.

Each layer interacts with the layers above and below. In the first case, it acts as a *service provider* making available a set of resources for the above *service user* layer. The interface between both layers defines how the available services can be accessed, how the service parameters must be handled, and how the response should be interpreted. The communication between the interfaces of layers N and N − 1 occurs via *service data units* (SDUs). The KNX specification also defines a rule set for the peer layer communication between devices. In this case, *protocol data units* (PDUs) are used, which mainly consist of user data and layer-specific *protocol control information* (PCI). In addition, four different communication primitives are defined: *request* (req), *indication* (ind), *confirmation* (con), and *response* (res). Services do not always need to use all the primitives and can be classified as in what follows:

- *Locally confirmed* services: require a request, an indication, and a confirmation. In this case, a request and the corresponding PDU are generated at layer N and passed to the lower layers until transmission on the physical medium. On the receiver side, the peer layer N is activated with an indication, the received PDU is progressively decoded, and the data are passed to the layers above. The sender layer N receives a confirmation from the local layer N − 1 indicating if the request has been processed correctly.
- *Confirmed* services: consist of a request, an indication, and a confirmation, but for these services, the peer remote layer generates an acknowledgment after receiving the indication. At the sender side, the received acknowledgment is passed to the local layer N as a confirmation.
- *Answered* services: always need a request, an indication, a response, and a confirmation. As shown in Figure 58.2, the response is generated by the remote service provider as a novel request issued toward the sender that will be received at the local layer N as a confirmation.

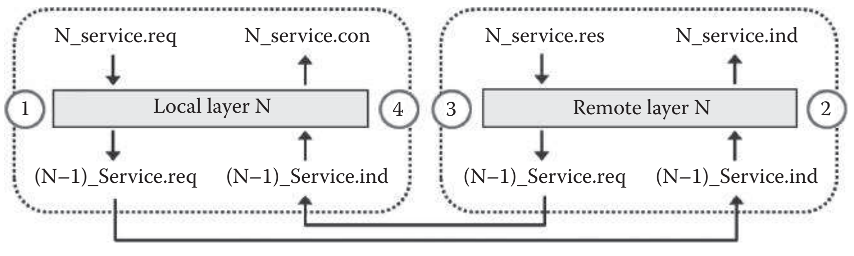


FIGURE 58.2 Answered service communication.

58.3.3 Network Architecture and Addressing Scheme

The KNX specification enables to build a fully distributed network, which accommodates up to 65,536 *devices* by exploiting a 16-bit individual address (IA) space. The logical topology or *subnetwork* structure allows 256 devices for each *line*. As shown in Figure 58.3, lines may be grouped together with a *main line* into an *area*. An overall domain is formed by 15 areas coupled with a *backbone line*. Not considering the addresses reserved for area and line couplers, up to 61,455 end devices may be joined by a KNX network.*

All components, that is, couplers or end devices, are unambiguously identified and subsequently accessed via their IA, reflecting the *area.line.device* topology. As shown in Figure 58.4, the IA consists of a two-octet value: the first byte defines the subnetwork address (SNA), while the last octet specifies the device address (DA) unique within a subnetwork. Routers always have a DA 0; other devices may have a DA with values ranging from 1 to 255. Devices needing no IA for operations may use a default IA (FF_h). This default IA shall consist of the DA for unregistered devices and the medium-dependent default SNA.

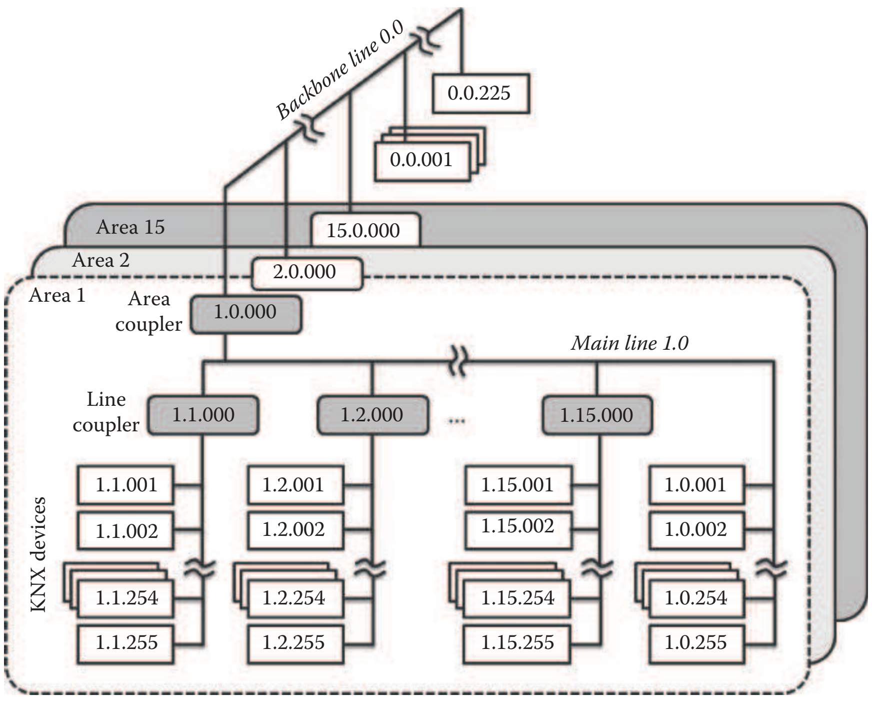


FIGURE 58.3 KNX network architecture.

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

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			Little Group							

FIGURE 58.4 KNX IA and GA.

* Installation and product guidelines should be taken into account. Possible restrictions may depend on implementation (e.g., medium, transceiver types, power supply capacity) and environmental factors (e.g., electromagnetic noise).

KNX also supports a full multicast addressing mainly used for the runtime communication, considerably reducing bandwidth requirements. The protocol provides a 16-bit group address (GA) space usually used in the format *main.middle.little* device groups as shown in Figure 58.4. Sometimes GA space can also be adopted with a two-*level* (main group/subgroup) or with a freely defined structure. The GA does not need to be unique. In fact, a device can also have more than one GA. Furthermore, each device is associated by default to the group zero (0/0/0) as request frames addressed to GA 0 will be sent in broadcast. In this way, a device is associated to the sender addresses with the aim of sharing its datapoints (i.e., group communication objects), thus creating a kind of network-wide variables. This communication model will be better described in Section 58.3.6. Since the protocol makes a 16-bit address space available to such GAs, it is possible to define about 2^{16} shared variables each with any number of local device instances.

In addition to IAs and GAs, if the network is built on an open medium (e.g., PL*) shared by devices of different subnetworks, a domain address (DoA) is required as an identifier to logically separate each subnetwork.

58.3.4 Frame Structure

Figure 58.5 shows the structure of a generic KNX frame built above the data link layer.

The control field is used to detect possible *repeated* frames (the flag R identifies a first transmission or already-sent frames), to indicate the frame *priority* (the flag PP can assume a *system*, *high*, *normal*, or *low* value), and to establish the frame *type* (standard or extended†). Then, there is an individual *source address*, in terms of area.line.device, and an individual or group *destination address* (respectively, in case of unicast or multicast communication). The destination address type is determined by a special field (*address type* [AT]) defined at the beginning of the network protocol data unit (NPDU). In addition to AT, the NPDU includes the *network layer protocol control information* (NPCI) field, a hop counter decreased by routers to avoid looping messages. NPCI ensures a frame can appear in seven physical network segments at most in order to avoid packets circulating endlessly due to errors in the network setup; when it becomes zero, the frame is silently discarded. The *length* field specifies the size of the NPDU, while the *transport layer protocol control information* (TPCI) field controls the communication at transport layer level, for example, specifying service codes and sequence numbers used to build up and maintain a point-to-point connection. Analogously, the *application layer protocol control information* (APCI) field contains the application layer service codes. The APCI has a length of 4 or 10 bits, depending on the specific service code. According to both addressing scheme and APCI, the standard frame can carry up to 14 *data* octets. Finally, the frame is closed by the *checksum*, which contains an odd horizontal parity value calculated over all preceding octets. This ensures data consistency and reliable transmissions.

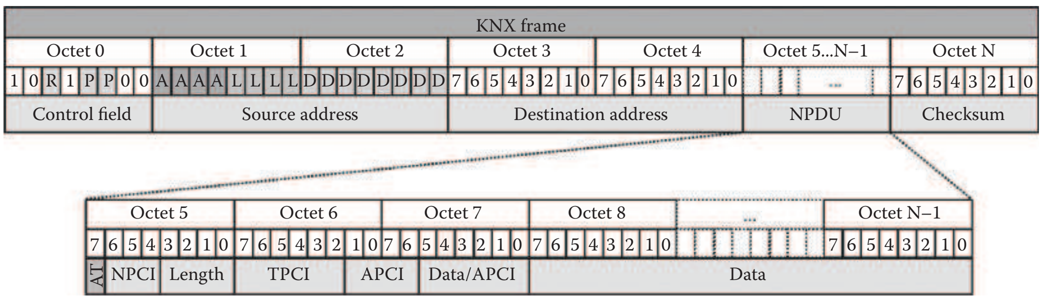


FIGURE 58.5 KNX frame structure.

* On a PL, nearby domains are logically separated with a 16-bit DoA.

† Despite the standard frame ensuring a direct backward compatibility toward the EIB protocol, an extended frame (identified by a "00" code in the two most significant bits) can be used to store up to 248 bytes of data.

58.3.5 Integration with IP Networks

The KNX specification describes a compact and flexible Internet protocol (IP) tunneling protocol, named *KNXnet/IP*, used to carry a KNX frame over an IP stretch enabling the communication of KNX implementations (e.g., KNXnet/IP devices) on the top of IP networks. In this way, several KNX subnetworks can be connected via an IP line acting as a fast (if compared to classic KNX transmission speeds) backbone. A widespread deployment of IP-based networks and applications is an opportunity to expand building control and communication beyond a local KNX network. Further benefits include (i) fast interface between local area network (LAN)-based and KNX-based systems and (ii) remote configuration and usage of home and building devices.

A KNXnet/IP system contains at least the following elements:

- A KNX subnetwork (e.g., KNX-TP1, KNX-RF, KNX-PL110) with up to 255 end devices.
- A KNX-to-IP network connection device (named KNXnet/IP *server*). It is a special device having a physical access to a KNX network also implementing the KNXnet/IP protocol to communicate with *clients* or other servers on an IP network channel. A server is by design also a KNX node with a unique IA.
- Additional software for remote operations hosted by client workstations (e.g., configuration tools, building management systems or browsers).

Figure 58.6 shows a basic network configuration where a KNXnet/IP client is connected to multiple KNX subnetworks via IP. The KNXnet/IP client may access one or more KNXnet/IP servers at a time.

When a KNXnet/IP server communicates with another server via IP, it acts also as a KNXnet/IP *router* exploiting a multicast one-to-many communication, where KNX data are simultaneously transferred from a sender device to one or more receivers over IP. A router can also replace a line or an area coupler and directly connect main lines and backbones, allowing the usage of existing cabling (e.g., Ethernet) and ensuring faster transmission times between KNX subnets.

A KNXnet/IP frame is a PDU moving along a non-KNX network. It contains a header, similar to the header of an IP data packet, and a body, if present. As shown in Figure 58.7, the header maintains information about the protocol version, the header length,* the total length of the packet, and the

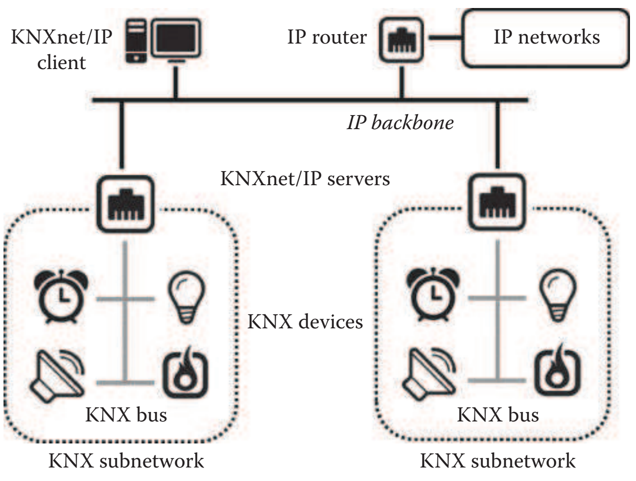


FIGURE 58.6 KNXnet/IP network example.

* Although the length of the header is always fixed, it is possible that it changes with newer versions of the protocol. The header length can be used as an index into the KNXnet/IP frame data to find the beginning of the KNXnet/IP body.

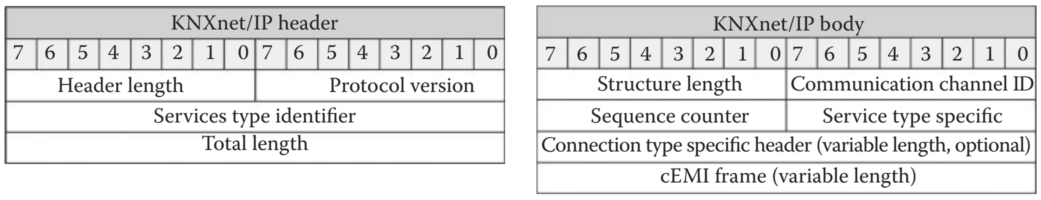


FIGURE 58.7 KNXnet/IP frame structure.

KNXnet/IP service-type identifier. KNXnet/IP services include, but are not limited to, information regarding device discovery, connection management, and KNX data transfer.

The header may be followed by a KNXnet/IP body of variable length, depending on a specific service. An example of a body is shown in Figure 58.7. It embeds the data into a basic KNX frame and starts with data fields that specify additional general information about the connection. Those fields compose the *connection header* whose appearance is fixed, varying only the content according to the connection options. Finally, the IP body includes the so-called cEMI frame consisting of the TP1 telegram structure, excluding the checksum field that is not used in IP communications as the error detection is done by IP.

58.3.6 Application Models

The *application interface layer*, as shown in Figure 58.1, is an additional abstraction level between the application layer and the user applications. It eases the communication tasks by offering a common interface that abstracts from many application layer details. It allows to expose shared objects and use them as local variables. In addition, it takes care of handling network management calls. The application interface layer contains the following objects:

- *Group objects* (GOs): they are accessible via application layer services in the multicast communication mode and provide a communication schema named *shared variable model*. GOs may also refer to one or more interface objects (IOs).
- IOs: they are accessible via application layer services in unicast and broadcast communication modes and are used for a *client/server* interaction. IOs are also classified as *system* IOs (e.g., device objects), relevant to network and device management, and *application* IOs, defined by the user applications.

The GO structure is shown in Figure 58.8. It consists of three parts: *description*, *value*, and *communication flags*. The GO description must at least include the GO *type* and the *transmission priority* (urgent, normal, or low). Optionally, the *config flags* include static information about the GO (e.g., read, write,

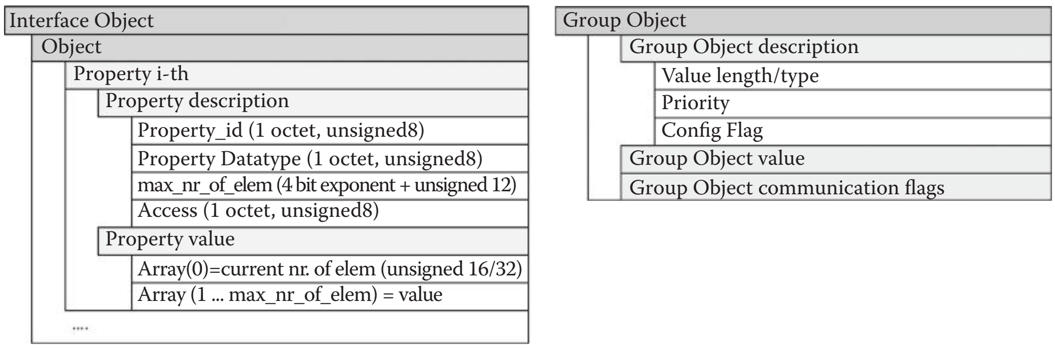

Interface Object
Object
Property i-th
Property description
Property_id (1 octet, unsigned8)
Property Datatype (1 octet, unsigned8)
max_nr_of_elem (4 bit exponent + unsigned 12)
Access (1 octet, unsigned8)
Property value
Array(0)=current nr. of elem (unsigned 16/32)
Array (1 ... max_nr_of_elem) = value
....
Group Object
Group Object description
Value length/type
Priority
Config Flag
Group Object value
Group Object communication flags

FIGURE 58.8 Interface and GO structure.

or update permissions). According to the GO type, the size of the value field can vary from 1 bit to 14 bytes. Finally, the communication flags show the *state* of a GO. The following states are enabled: *update, read_request, write_request, transmitting, ok, and error.*

The IOs are instances of the general structure shown in Figure 58.8. Starting from this common schema, two object types are derived: *full* IOs and *reduced* IOs.

Full IOs exactly comply with the data structure reported in Figure 58.8. They consist of a number of *object properties* composed by a *description* and a *value* field. In turn, a property description consists of the following fields:

- Property index: it is unique for each IO. The first property has index 0, and further ones are numbered with subsequent values without gaps.
- Property IDentifier (PID): this value is encoded in the *property_id* field and is usually used to identify the specific IO.
- Property data type (PDT): it describes the most appropriate data type for the IO property.
- Maximum number of elements: the value of a property is always an array, so this field specifies the array size: suitable indexes range from 1 to *max_nr_of_elem*, while the element at position 0 contains the current number of valid array elements. This value is automatically updated if an element is written beyond the current last element but always within the maximum allowed number of entries.
- Read/write access levels: this attribute indicates the access level needed to read or write the property value.

Moreover, each property description has also a *write-enable flag* that specifies whether the property value can be written or not. In particular, the property with *property_id* equals to 1, and *index* 0 is named *IO type* (`PID_OBJECT_TYPE`) and contains the description of the IO itself. This property is mandatory for every IO.

As opposed to full IOs, reduced IOs only support a subset of the common IO structure. A property description is only composed of the property_id, while the property value can be a single value or an array.

58.4 Development and Configuration

On the top of KNX networks, several distributed applications can be built, enabling a tight interaction among different devices. This section summarizes some distinctive characteristics of the KNX standard at the application level. Modeling guidelines and interworking models are resumed to give a sketch of a comprehensive and multidomain building communication system. Hence, device configuration modes and development tools are detailed and a basic configuration example of a KNX system is provided.

58.4.1 Interworking Specification

The interworking between devices enables different and multivendor products to send and receive datagrams and properly understand and react on them without additional equipments (e.g., gateways). In this, devices from different manufacturers can interwork in a given application domain as well as across different applications. Interworking is a relevant feature of the KNX standard: products not complying with this specification cannot obtain the KNX certification. In order to establish this communication model, a set of requirements was defined, grouped into the *application interworking specification* (AIS). AIS is defined for various application domains and can be further extended by submitting proposals to the KNX Association. In fact, the application *modeling* (the process of analyzing and defining the model for each application) is the responsibility of various application *specification groups* of the KNX Association.

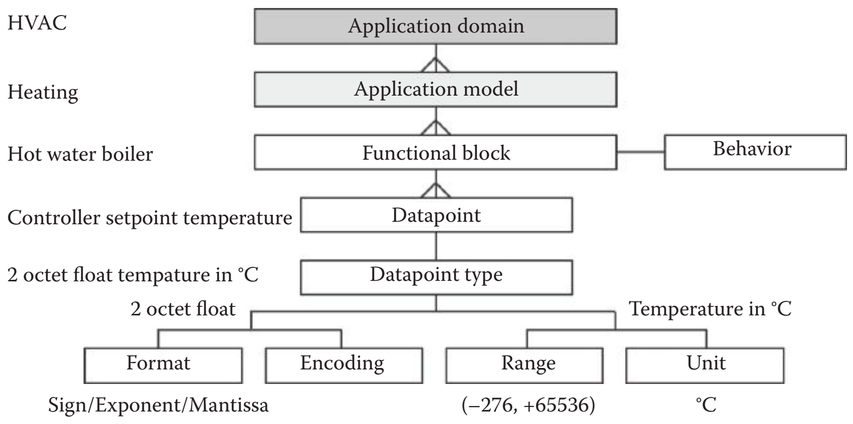


FIGURE 58.9 Example of an application modeling.

As presented in Figure 58.9, each application domain encompasses one or more applications, each of them defined in terms of *functional blocks* (FBs) that communicate with each other. In this way, a *distributed* application is composed of a number of shared FBs implemented in various devices in the network. In addition, a manufacturer may group one or more FBs, of the same, or of different applications, to build a device. An FB is described by the product developer as an object with well-defined behaviors; it uses one or more *datapoints* for interchanging information, which represent inputs, outputs, parameters, or diagnostic data. A datapoint can be seen as a generic interface used to set, receive, or transmit data during the runtime operations.

The information exchanged through datapoints is associated to a *DataPoint Type* (DPT) featuring the data in terms of the following:

- *Format*: describes both sequence and length of the fields composing the DPT.
- *Encoding*: indicates how the data are encoded according to the given format.
- *Range*: describes the limitation of the values that may be encoded in the DPT. This may be a minimum/maximum indication or an explicit list of values.
- *Unit*: specifies the measurement unit of the carried information.

The DPTs are identified by a 16-bit *main number* separated by a dot from a 16-bit *subnumber* (e.g., 7.002). The main number identifies DPTs with the same format and encoding, whereas the subnumber indicates a specific dimension (different range and/or unit).

58.4.2 Network Configuration

The KNX standard allows each manufacturer to select the most proper configuration mode according to different markets, local habits, level of training needed, or application environment. In particular, four different *configuration modes* are available:

1. *System* (S) mode (S-mode): the most versatile and multiusage configuration process. The overall application setup is done by a configuration master. Usually, it is done by means of a set of PC-based tools from the engineering tool software (ETS) family, developed by the KNX Association. With the aid of the ETS tools, described in Section 58.4.3, it is possible to configure the S-mode compliant devices and set them into operation.
2. *Controller* (Ctrl) mode: an external controller device is required to set the IAs and the needed parameters in the devices. The controller is not needed after configuration but may also have additional runtime functionalities.

3. *Push-button* (PB) mode: no tools or external devices (e.g., PC, controller, ETS) are needed for the configuration and linking management. Devices themselves are able to set up the links and assign the IAs and GAs. However, to enable the development and manufacturing of low-cost products, the software overhead for the mandatory configuration and link procedures must be as small as possible.
4. *Logical tag extended* (LTE) mode: mainly designed to cover the specific needs of an easy configuration for HVAC, which needs a longer set of structured data. These data are exchanged via IOs using the extended frame of the KNX protocol. Also in this case, a simple device configuration is possible without using PC tools, manufacturer specification, or the knowledge of device object structures and GAs.

In order to facilitate the network design and configuration and to simplify the device certification, previous configuration modes have been grouped in so-called configuration profiles. The KNX profiles have been designed to cover all needs and habits of the KNX Association community. In particular, a profile specifies how a device should be configured and the minimal device requirements for this process. The KNX specification defines the following profiles:

- *S-mode* profiles: suitable for end devices and system components (i.e., couplers) supporting the S-mode configuration. This type of setting procedure is adopted by well-trained KNX installers in order to realize advanced building control functions enabled via ETS tool using specific product databases provided by device manufacturers. Each database includes all supported functionalities and ETS is also used to set the device parameters as required by the installation. S-mode offers the highest degree of flexibility for building KNX networks.
- Easy mode *(E-mode)* profiles: adopted for devices supporting the controller, PB, and LTE configuration modes. The installation of such components can be also made by users with a basic KNX experience. E-mode compatible products offer limited features compared to S-mode. In this case, components are usually preprogrammed and loaded with a default set of parameters. With a simple configuration software, the device parameter settings and communication links can be partially reconfigured.

58.4.3 Development Tools

In order to plan, design, and commission a KNX installation, a software tool is required to assist designers and electrical installers. For that reason, a manufacturer-independent configuration tool named ETS was developed by the KNX Association. ETS4 [8] is the current release of the tool, replacing since 2010 ETS3. During its development, the user interface has been basically kept very similar to the one of previous releases. On the other hand, both functions and operating philosophy of the ETS4 user interface has been completely reengineered, making them simplified compared to ETS3. ETS is available in three different versions:

1. *Professional*: which enables the development of KNX installations of any size without limits in terms of maximum number of addresses, devices, or project.
2. *Light*: maintaining the same functions as the professional version; this version only allows projects with up to 20 devices. It is intended for training centers and their customers.
3. *Demo*: this version is free of charge but can be used to design networks with three devices at most.

In case of a user with a single PC-dependent license, there is also a *supplementary* version available as an additional license for the full professional version running on a second PC, for example, a laptop used at the construction side. In addition to ETS, the KNX Association also offers the following software tools or libraries:

- *iETS server*: it is a software interface between KNX and IP adopted for connecting iETS clients (such as ETS4) to the KNX network. With iETS, the installer can access the system remotely by means of ETS, enabling remote programming or diagnostics via IP.
- *Falcon Driver Library*: it is a high-performance DCOM (Microsoft Distributed Component Object Model) based library enabling Windows-based PC KNX bus access. Falcon offers an API for sending and receiving telegrams across the KNX network and supports the communication through RS232, USB, and Ethernet.
- *Interworking test tool* (*EITT*): this tool is used by developers and test laboratories for testing, troubleshooting, and monitoring the interworking and system stack compliance of KNX products. EITT is also a powerful tool for the analysis and simulation of the KNX network protocol.

Also KNX members offer a number of KNX-related software packages. In particular, several ETS plug-in modules (named ETS Apps) have been developed, allowing to extend the functionality of ETS, for example, for graphical project designs, data exchange, or efficient project engineering. In fact, any new, or existing, software can be adapted to the ETS App interface and integrated in ETS4 by using the related SDK (software development kit), without the need to completely recompile the software.

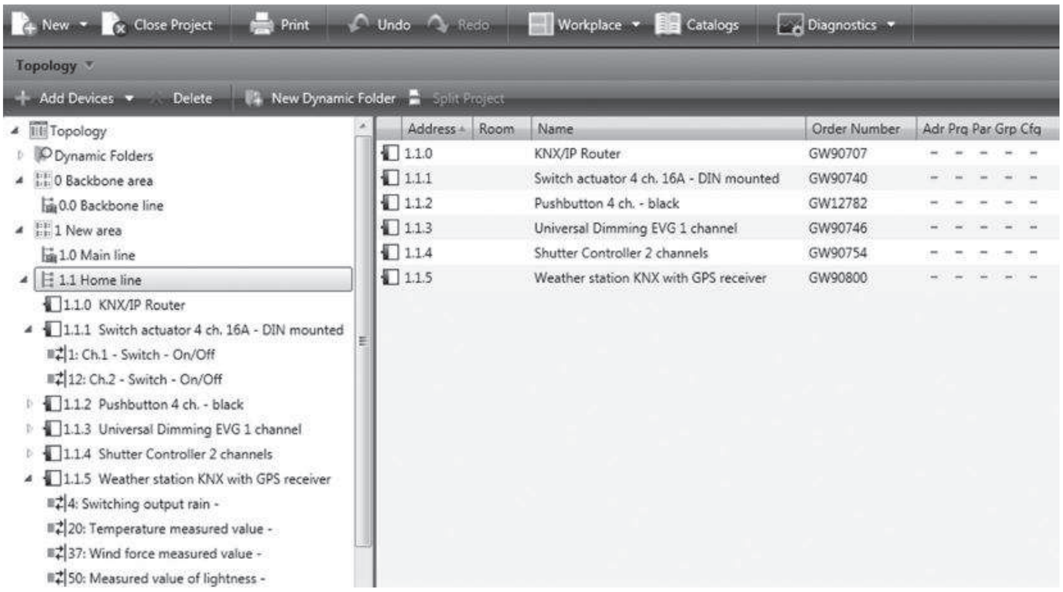
Finally, noteworthy is a Java library for KNX access, named *Calimero* [9], offering a clean and lean interface and allowing free client access to KNX networks and a basic KNX server functionality. It is designed to operate also on resource-constrained systems (i.e., mobile and embedded devices) as it requires low memory and CPU. In addition, it is fully compliant with the popular Java Standard Edition (J2SE) environment commonly deployed on home computers, and this ensures a widespread employment of it. It is important to notice that Calimero is an open-source software distributed according to the GNU GPL license.

58.4.4 Basic Installation

In order to illustrate the process of setup and configuration of a KNX network, a simple example is considered. The project design of a KNX system initially does not differ from a typical electrical project. In the preliminary stages, several aspects must be clarified by the planner such as the type and use of the installation, the building system components to use, and their functions and special requirements stipulated by customers. Subsequently, product databases may be downloaded and imported in ETS4 in order to create a new project. Then the network configuration must be defined, for example, building structure, bus topology, and GA levels. The project engineer selects the KNX devices with the corresponding application to insert into the network, assigns them an IA, and sets the parameters according to the requested features. As an example, a network is proposed consisting of a simple TP line (1.1) connected to a backbone IP line (0.0), having six devices connected: a KNX/IP router, a switch and a dimmer actuator, a KNX module with PBs, a shutter controller, and a weather station for outdoor environments (Figure 58.10). The IAs are automatically assigned by ETS4 in ascending order in the respective current line; however, they can be modified at any time by the user. The selected devices are displayed in the tree structure on the left panel. Double-clicking on each device, the properties panel will be displayed containing further information about the product and the parameters that can be modified by the installer. Moreover, selecting a device in the tree view, the related GOs will be shown below it. In the proposed configuration, the weather station (1.1.5) exposes several GOs associated to rain, temperature, wind force, and brightness sensors.

Afterward, GAs can be created. In each project, a different pattern for the definition of GAs can be used. For example, for three-level addresses, the main group can specify floors or rooms, the middle group can specify the functional domains (e.g., switching, dimming), and the subgroup can define articulated operations named scenarios (e.g., turn on/off a specific lamp in the house or open/close the living room according to the weather conditions). The GAs are listed in the GA window, shown in Figure 58.11. In order to ensure that sensor and actuator devices may know which GOs should communicate with each other to realize a given function, the objects of each device are assigned to the GAs.* In the example, the rain sensor of the weather station is connected to the shutter controller by means of the same GA, so when rain is detected, the shutters will be automatically closed.

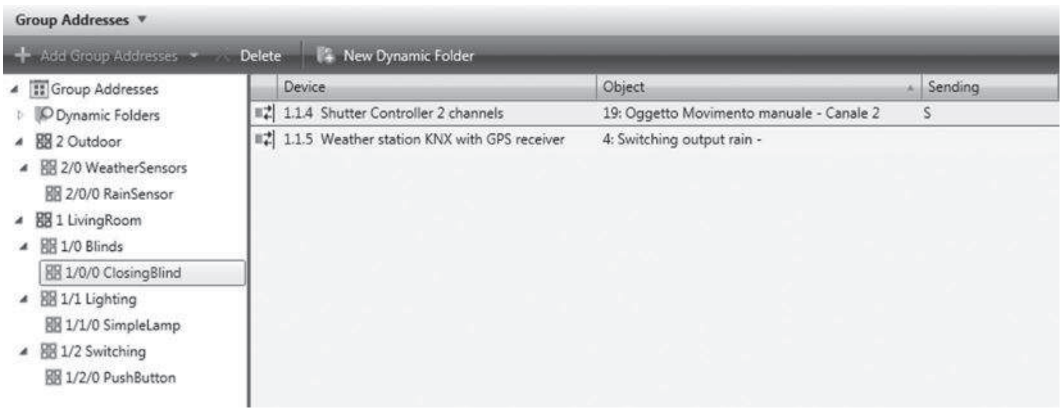
In this way, all devices are fully configured and the final network topology is defined. In conclusion, the network configuration should be checked before the commissioning of the project. Obviously, it is also possible to deviate from this sequence according to the project requirements. Some steps can be omitted for smaller projects, whereas additional steps could be needed in larger ones.



```
Topology
  Dynamic Folders
  0 Backbone area
    0.0 Backbone line
  1 New area
    1.0 Main line
    1.1 Home line
      1.1.0 KNX/IP Router
      1.1.1 Switch actuator 4 ch. 16A - DIN mounted
        1: Ch.1 - Switch - On/Off
        12: Ch.2 - Switch - On/Off
      1.1.2 Pushbutton 4 ch. - black
      1.1.3 Universal Dimming EVG 1 channel
      1.1.4 Shutter Controller 2 channels
      1.1.5 Weather station KNX with GPS receiver
        4: Switching output rain -
        20: Temperature measured value -
        37: Wind force measured value -
        50: Measured value of lightness -
```

Address	Room	Name	Order Number	Adr	Prg	Par	Grp	Cfg
1.1.0		KNX/IP Router	GW90707	-	-	-	-	-
1.1.1		Switch actuator 4 ch. 16A - DIN mounted	GW90740	-	-	-	-	-
1.1.2		Pushbutton 4 ch. - black	GW12782	-	-	-	-	-
1.1.3		Universal Dimming EVG 1 channel	GW90746	-	-	-	-	-
1.1.4		Shutter Controller 2 channels	GW90754	-	-	-	-	-
1.1.5		Weather station KNX with GPS receiver	GW90800	-	-	-	-	-

FIGURE 58.10 ETS4 topology panel.



```
Group Addresses
  Dynamic Folders
  2 Outdoor
    2/0 WeatherSensors
      2/0/0 RainSensor
  1 LivingRoom
    1/0 Blinds
      1/0/0 ClosingBlind
    1/1 Lighting
      1/1/0 SimpleLamp
    1/2 Switching
      1/2/0 PushButton
```

Device	Object	Sending
1.1.4 Shutter Controller 2 channels	19: Oggetto Movimento manuale - Canale 2	S
1.1.5 Weather station KNX with GPS receiver	4: Switching output rain -	

FIGURE 58.11 ETS4 GA panel.

* In a group address, it is only possible to create connections between group objects of the same type (e.g., 1 bit, 1 byte). The group address receives the type of the group objects with the first assignment.

58.5 Research Perspectives

The basic goal of HBA [10] is to improve people's comfort, increase safety and security, reduce energy consumption, and minimize the overall ecological footprint. For this purpose, a more effective management and coordination of building appliances and subsystems is required. In particular, *ambient intelligence* (AmI) [11] proposes a multidisciplinary research vision aiming at defining frameworks and techniques for smart environments, where people are surrounded by intelligent and unobtrusive microcomponents, capable of being sensitive and responsive, recognizing user profiles, and self-adapting behavior accordingly. Devices should communicate and interact autonomously, without the need for direct user intervention, also making decisions based on multiple factors, including contextual information, sensory data patterns, and user presence and preferences. They should be coordinated by intelligent systems acting as supervisors.

At present, EIB/KNX and other current standard HBA systems and technologies are still far from that vision, because they cannot grant such flexibility. As seen in previous sections, they can essentially set static functions and features configured during system setup; explicit interaction with the user is required for all except the most elementary cases. Context-aware dynamic information management and decision-making about users, devices, and services are not yet supported. In order to enable intelligent HBA infrastructures, capable to adapt and control building appliances autonomously, smart environments have to be conceived according to results coming from pervasive and mobile computing, artificial intelligence theory, and agent-based software design [12]. Consequently, AmI research is closely related to studies for effective discovery and coordination in volatile and resource-constrained scenarios.

In latest years, four major research directions were pursued to improve intelligence, effectiveness, and usability of HBA systems:

1. Improve user interaction with HBA systems.
2. Integrate different HBA technologies in a common application layer.
3. Model HBA frameworks as *multiagent systems (MASs).*
4. Use semantic technologies to enhance information management and decision-making capabilities.

1. Several approaches proposed more advanced and user-friendly device management with respect to current standard technologies. Nevertheless, they still rely only on direct command-based user interaction, through local or network-based interfaces. In particular, [13] introduced a method to control home devices either locally (through an AJAX web application) or remotely (via web services), allowing the user to issue either direct commands to devices or set routines that should be executed periodically. The proposal is not tied to any particular HBA standard. The framework devised in [14] is focused on a low-cost ZigBee home automation system. Also in this case, devices can be controlled directly by the home occupants by means of a ZigBee remote control, while the ZigBee *Home Automation* network exposes a *home gateway* on the Internet, so that the home can also be controlled remotely from a PC.
2. Due to the diversity of HBA technologies, a further aspect widely investigated is the integration of different HBA technologies in a common application layer. Some works achieved limited integration of domotic subnetworks based on different technologies by means of custom gateways. For example, in [15], a gateway is proposed between KNX and ZigBee *Home Automation* application profile. Several proposals [16,17] adopt web service technologies as unified abstraction layer to integrate various communication protocols and mobile service discovery via UDDI (Universal Description, Discovery, and Integration) and UPnP (Universal Plug and Play) in automation contexts. Standard web service description and orchestration languages based on XML are adopted, only allowing syntactic match between users' needs and service/resource attributes, lacking semantic characterization. As a consequence, the improvement in service discovery and composition capabilities with respect to basic HBA protocols is not so relevant to justify the increased architectural complexity.

3. HBA is characterized by the lack of hard real-time constraints and the need for high flexibility and reconfiguration capabilities. As pointed out in [18], such kinds of environments are very well suited to be modeled by means of MASs. Agents can meaningfully represent entities (devices), contexts (rooms, floors), or people, emphasizing interaction capabilities such as communication, concurrency, cooperation, coordination, conflict resolution, and negotiation. Many proposals exist in literature for structuring HBA systems through MAS: while early research focused on agent-based direct control of home appliances, recent MAS proposals concern high-level, complete HBA control systems. In order to achieve greater interoperability, a major goal is to provide a unified view of the home environment and a flexible application model. Particularly, in [19], a platform was proposed to allow an easy integration of multimodal user interfaces in heterogeneous device networks. The MAS was implemented using Java Agent Development Framework (JADE) SDK, whereas IP-based UPnP was exploited as discovery protocol. In [20], a service-oriented smart home architecture was defined, where each component is designed as an agent. When the home is about to perform a service for a user, it compares service requirements with the environment situation to find out rooms whose status and resources are ready for service activation. Similarly, in [21], the use of BDI (belief–desire–intention) agents was proposed to automate service composition tasks, providing transparency from the user standpoint. In latest years, the MAS model has been proposed also for *BEMS* (*Building Energy Management Systems*), due to the growing interest in *smart grid* technologies [22] and in reducing energy consumption via *demand-side management* [23]. An agent-based BEMS typically supports *active load management* techniques [10] and exploits *smart metering* via energy-efficient network protocols [24]. Such approaches *put the human back into the loop* [23], that is, they take into account user profiles and requests in order to maximize both energy efficiency and customer satisfaction. In [25], a *multiagent comfort and energy system* was designed for management and control of both building systems and occupants. It coordinates appliances and users, for example, suggesting changes to occupant meeting schedules. A MAS for energy management in commercial buildings was proposed also in [26]: Agents communicate and negotiate with human occupants to save energy; optimal policies are generated considering multiple criteria—for example, energy cost and personal comfort—as well as uncertainty in occupant preferences.
4. Knowledge representation and reasoning (KRR) technologies allow greater generality and more flexible reuse of models concerning the aforementioned approaches, because *ontologies* provide a conceptual framework to express and share formal and structured descriptions of services and appliances, while general-purpose reasoning procedures can be used for semantic-based service composition in different HBA scenarios. *DomoML* [27] was the first specific proposal of a building automation ontology suite. Reinisch et al. [28] acknowledged the relevance of semantic-enhanced approaches upon current HBA standards, for cost and efficiency motivations; they introduced a theoretical ontology-based framework for the integration of different HBA protocols at the application level. In [21], the use of intelligent agents was proposed to automate service composition tasks. Nevertheless, a very elementary ontology was derived from attribute-based service descriptions in UPnP and Bluetooth service discovery protocols. As a consequence, the approach lacked adequate expressiveness of user, device, and service profiles. In [29], the first self-contained prototype was presented including a reasoning module able to manage and coordinate heterogeneous devices by means of logic rules processing. Classical rule-based inferences are not enough in heterogeneous and dynamic AmI contexts. In order to execute a rule, conditions it imposes must be fully matched by the current system state. Unfortunately, experience shows that full matches are quite unlikely in real-life scenarios, where objects, subjects, and events are featured by different heterogeneous descriptions, often partially in conflict among them. Semantic-based matchmaking frameworks as the one in [30], which exploit standard and nonstandard inference services and allow to match requests and resources based on the meaning of their descriptions (also providing classification and

logic-based ranking), are more effective. Beyond obviously good matches, such as *exact* or *full* ones, they enable so-called *potential or intersection* matches (i.e., those matches where requests and supplied resources have something in common and no conflicting characteristics) and *partial or disjoint* matches (i.e., cases where requests and available services have some conflicting features) that can also be considered useful in scenarios when nothing better exists. In [31], the exploitation of KRR technologies, originally conceived for the semantic web, was proposed to overcome restrictions of common HBA systems. An enhancement to ISO/IEC 14543-3 EIB/KNX standard has been devised in a knowledge-based and context-aware computing framework for building automation, supporting semantic annotation of both user profiles (i.e., needs, moods, features) and device capabilities (i.e., services/resources of home appliances). The integration of a semantic microlayer within the KNX protocol stack enabled novel resource discovery and decision support features in HBA making them autonomous and decentralized while preserving full backward compatibility. Machine-understandable metadata characterize both home environment and user profiles and preferences. Thanks to the integration of semantics at the application layer, each object/subject joining a KNX network can describe itself and advertise managed services/resources. By means of a matchmaking process—based on inference procedures in [30]—the most suitable available services/functionalities for adapting the ambient to a given request or event can be easily detected. Annotations are expressed in ontological formalisms derived from description logics (DLs) [32]: DIG [33], a more compact equivalent of OWL-DL (OWL Web Ontology Language, W3C Recommendation, February 10, 2004, http://www.w3.org/TR/owlfeatures/), has been adopted in particular. In order to reduce the size of semantic annotations referred to both device features and user profiles, an encoding algorithm [34] is exploited for efficiently compacting XML-based ontological languages, generating a header and a body for each encoded document. Furthermore, an IO named *generic profile of the device* (*GPD*) has been introduced to describe general device features, for example, type, manufacturer, or model. A single GPD is associated to a given device. A *specific profile of the device* (*SPD*) object has been also defined to describe individual functionalities and operating modes of a device. Multiple SPDs can be associated to the same device, one for each different service/function it exposes. Furthermore, a user-transparent and device-driven interaction is enabled as opposed to current static configuration approaches. At network layer, KNX support for IP communication through KNXnet/IP is leveraged to extend the management of building control beyond the local bus, while IEEE 802.11 and Bluetooth are exploited for wireless communication with the user. The work was extended in [35] with a multiagent approach exploiting semantic-based resource discovery and orchestration in HBA. The semantic annotation of user profiles and device capabilities is used to (1) determine the most suitable home services/functionalities according to implicit and explicit user needs and (2) allow device-driven interaction for autonomous adaptation of the environment to context modification.

References

1. UN1 EN 15232, Energy performance of buildings—Impact of building automation, controls and building management, 2007.
2. KNX Association, *KNX Handbook for Home and Building Control*, KNX, Brussels, Belgium, 2009.
3. EN 50065-1, Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148,5 kHz—Part 1: General requirements, frequency bands and electromagnetic disturbances, 2011.
4. ISO/IEC 8802-2, IEEE Standard for Information Technology, Telecommunications and information exchange between systems—Part 2: Logical link control, Washington, DC, 1998.
5. ISO/IEC 8802-11, IEEE Standard for Information Technology, Telecommunications and information exchange between systems—Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 2012.

6. IEEE 1394.1, IEEE standard for high performance serial bus bridges, 2004.
7. ISO/IEC 7498-1, Information technology—Open systems interconnection—Basic reference model: The basic model, 1994.
8. KNX Association, ETS4 (Engineering Tool Software), version 4, 2010. http://www.knx.org/knx-en/software/ets/about/index.php.
9. B. Malinowsky, G. Neugschwandtner, W. Kastner, Calimero: Next generation, in *Proceedings of the KNX Scientific Conference*, Duisburg, Germany, 2007.
10. D. Dietrich, D. Bruckner, G. Zucker, P. Palensky, Communication and computation in buildings: A short introduction and overview, *IEEE Transactions on Industrial Electronics*, 57(11), 3577–3584, IEEE, Washington, DC, 2010**.**
11. N. Shadbolt, Ambient intelligence, *IEEE Intelligent Systems*, 18, 2–3, IEEE, Washington, DC, 2003.
12. H. Nakashima, H. Aghajan, J.C. Augusto, *Handbook of Ambient Intelligence and Smart Environments*, Springer, New York, 2010.
13. M.E. Pardo, G.E. Strack, D.C. Martinez, A domotic system with remote access based on web services, *Journal of Computer Science & Technology*, 8(2), 91–96, 2008.
14. K. Gill, S. Yang, F. Yao, X. Lu, A ZigBee-based home automation system, *IEEE Transactions on Consumer Electronics*, 55(2), 422–430, IEEE, Washington, DC, 2009.
15. W.S. Lee, S.H. Hong, KNX—ZigBee gateway for home automation, in *IEEE International Conference on Automation Science and Engineering* (*CASE 2008*), Arlington, VA, 2008, pp. 750–755.
16. V. Miori, L. Tarrini, M. Manca, G. Tolomei, An open standard solution for domotic interoperability, *IEEE Transactions on Consumer Electronics*, 52(1), 97–103, 2006.
17. T. Catarci, F. Cincotti, M. Leoni, M. Mecella, G. Santucci, Smart homes for all: Collaborating services in a for-all architecture for domotics, in *5th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Washington, DC, 2009, pp. 56–69.
18. M. Metzger, G. Polaków, A survey on applications of agent technology in industrial process control, *IEEE Transactions on Industrial Informatics*, 7(4), 570–581, 2011.
19. K.I.-K. Wang, W.H. Abdulla, Z. Salcic, Ambient intelligence platform using multi-agent system and mobile ubiquitous hardware, *Pervasive and Mobile Computing*, 5, 558–573, 2009.
20. C.-L. Wu, L.-C. Fu, Design and realization of a framework for human-system interaction in smart homes, *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(1), 15–31, IEEE, Washington, DC, 2012.
21. M. Santofimia, F. Moya, F. Villanueva, D. Villa, J. Lopez, Intelligent agents for automatic service composition in ambient intelligence, in *Web Intelligence and Intelligent Agents*, Z. Usmani, ed., InTech, Rijeka, Croatia, pp. 411–428, 2010.
22. V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, G. Hancke, Smart grid technologies: Communications technologies and standards, *IEEE Transactions on Industrial Informatics*, 7(4), 529–539, IEEE, Washington, DC, 2011.
23. P. Palensky, D. Dietrich, Demand side management: Demand response, intelligent energy systems, and smart loads, *IEEE Transactions on Industrial Informatics*, 7(3), 381–388, IEEE, Washington, DC, 2011.
24. M. Aliberti, Green networking in home and building automation systems through power state switching, *IEEE Transactions on Consumer Electronics*, 55(2), 445–452, IEEE, Washington, DC, 2011.
25. L. Klein, J. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham, M. Tambe, Coordinating occupant behavior for building energy and comfort management using multi-agent systems, *Automation in Construction*, 22, 525–536, 2012.
26. J. Kwak, P. Varakantham, R. Maheswaran, M. Tambe, F. Jazizadeh, G. Kavulya, L. Klein, B. Becerik-Gerber, T. Hayes, W. Wood, SAVES: A sustainable multiagent application to conserve building energy considering occupants, in *11th International Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), Valencia, Spain, 2012, pp. 21–28.

27. L. Sommaruga, A. Perri, F. Furfari, DomoML-env: An ontology for human home interaction, in *Proceedings of the 2nd Italian Semantic Web Workshop* (*SWAP'05*), *CEUR Workshop*, Trento, Italy, December 2005, pp. 249–255.
28. C. Reinisch, W. Granzer, F. Praus, W. Kastner, Integration of heterogeneous building automation systems using ontologies, in *Proceedings of the 34th Annual Conference on Industrial Electronics* (*IECON'08*), Orlando, FL, 2008, pp. 2736–2741.
29. D. Bonino, E. Castellina, F. Corno, The DOG gateway: Enabling ontology-based intelligent domotic environments, *IEEE Transactions on Consumer Electronics*, 54(4), 1656–1664, IEEE, Washington, DC, November 2008.
30. S. Colucci, T. Di Noia, A. Pinto, A. Ragone, M. Ruta, E. Tinelli, A non-monotonic approach to semantic matchmaking and request refinement in e-marketplaces, *International Journal on Electronic Commerce*, 12(2), 127–154, 2007.
31. M. Ruta, F. Scioscia, E. Di Sciascio, G. Loseto, Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation, *IEEE Transactions on Industrial Informatics*, 7(4), 731–739, IEEE, Washington, DC, November 2011.
32. F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, P. Patel-Schneider, *The Description Logic Handbook*, Cambridge University Press, Cambridge, U.K., 2002.
33. S. Bechhofer, R. Möller, P. Crowther, The DIG description logic interface, in *Proceedings of the 16th International Workshop on Description Logics* (*DL'03*), ser. *CEUR Workshop Proceedings*, vol. 81, Rome, Italy, September 2003.
34. F. Scioscia, M. Ruta, Building a semantic web of things: Issues and perspectives in information compression, in *IEEE International Conference on Semantic Computing* (*ICSC'09*), Berkeley, CA, 2009. IEEE, Washington, DC, pp. 589–594.
35. M. Ruta, F. Scioscia, G. Loseto, E. Di Sciascio, Semantic-based resource discovery and orchestration in Home and Building Automation: A multi-agent approach, *IEEE Transactions on Industrial Informatics*, 10(1), 730–741, IEEE, Washington, DC, February 2014.