Communication Services for Secure Building Automation Networks

Wolfgang Granzer and Wolfgang Kastner
Vienna University of Technology, Institute of Computer Aided Automation, Automation Systems Group
Treitlstraße 1-3, A-1040 Vienna, Austria
Email: {w, k}@auto.tuwien.ac.at

***Abstract*—Up to now, building automation systems were considered as virtually closed enviroments. If at all, they had to provide some well dedicated dial-in connections for remote management. With the introduction of interconnections to foreign networks (e.g., Web gateways to the Internet) and wireless technologies, the assumption of physical isolation is not longer valid. In parallel, building automation systems of the next generation shall also be home for tight integrated services with seamless interworking nodes of formerly separated systems. When security-critical integration is considered, this promises synergies, but significantly tightens requirements on the protocol stack. This paper summarizes the demands to be met by nodes participating in such an environment and presents necessary secure services that are part of the protocol architecture.**

I. Introduction and Motivation

Building automation comprises all aspects of services for controlling and managing a functional building. Core domains are still the high energy intensive areas of Heating, Ventilation, and Air-Conditioning (HVAC), and lighting/shading. In the early years of building automation, realization of communication structures for device interconnection focused on a domain specific centralized approach [1]. Reasons were the limited capabilities of the involved devices, mainly restricted to their actual domain intrinsic functionality (e.g., sensing or actuating for room temperature control, constant light control). The technical evolution of field devices and of communication systems led to more intelligent sensors, actuators, and controllers as well as networks segments to connect them. To force data exchange inside a specific domain, network segments were connected via a common backbone where meanwhile the Internet Protocol (IP) found a widespread acceptance.

With the (still) ongoing down-shift of intelligence, devices are nowadays capable of processing more tasks locally without need of a central controlling instance. As a future trend, decentralized considerations for communication structures will even allow sensor sharing and sensor fusion across a domain. On this way, further services from the safety and security domain may be fully integrated in an automated building. However, when such services are going to be integrated, it is required to keep sensitive information confidential and prevent abuse of the resulting system for economic or even life safety reasons [2].

Available technologies do not satisfy the requirements for an integration of security-critical applications – some solutions provide a solid base (e.g., BACnet, IEEE 802.15.4/ZigBee), while others at least include rudimentary features (e.g., KNX, LonTalk) [3]. After identifying the most important requirements for a secure communication (Section II), this paper presents a generic architecture that allows a secure exchange of data between security-critical control applications (Section III). Resting upon existing low-level protocol mechanisms, a multi-protocol communication stack is defined (Section IV, V, and VI). Accompanied by specific quality of service features, this stack supports secure communication services that provide a common environment for security-critical building automation systems. The paper is concluded by an outlook on implementation details (Section VII).

II. Requirements

A. Security

To be able to secure communication between all kinds of building automation applications, the following security objectives have to be met[1].

- *Entity authentication*: Guaranteeing entity authentication avoids that a malicious node impersonates a trustworthy, legitimate one. While in most cases both the receivers as well as the senders must prove their identities (*mutual entity authentication*), it may be sufficient that only one side i.e., either the receiver side or the sender side within a secure relationship is authenticated (*unilateral entity authentication*). A typical example of a relationship where only unilateral entity authentication is demanded is a sensor that periodically broadcasts data. In such a case, it is important that the identity of the sender is authenticated – proving the authentication of the receivers is not required.
- *Authorization*: After the authentication of the members of a secure communication relationship has taken place, it must be verified whether the joining node has the necessary access rights to attend a relationship. If it has insufficient access rights, participating in a relationship must be denied.
- *Secured channel*: After the members of the relationship are authenticated and all of them have the required access rights, the data that is exchanged within the relationship must be protected against security attacks. This is done by establishing a so called *secured channel*. A

[1]The used security terms and definitions are based on [4].

978-1-4244-6391-6/10/$26.00 ©2010 IEEE

secured channel uses non-cryptographic (e.g., physical or organizational measures) and/or cryptographic techniques (e.g., using Message Authentication Codes (MACs) or encryption algorithms) to protect data against security attacks while it is transmitted over an insecure network. Depending on the requirements of the involved nodes, a secured channel may guarantee:

- *Data integrity*. Providing data integrity guarantees that the data was not modified by unauthorized nodes during transmission. To achieve this, modification attacks have to be prevented. However, if a full prevention is not possible, modification attacks shall at least be detected in order to avoid the use of corrupted data.
- *Data origin authentication*. In addition to data integrity i.e., the protection of data against unauthorized modification, a receiver can uniquely identify the data origin i.e., the data source.
- *Data freshness*. Data freshness guarantees that the transmitted data is recent and that an adversary has not replayed previously sent data. A key feature of guaranteeing data freshness is message ordering.
- *Data confidentiality*. The disclosure of confidential information must be avoided. It must be guaranteed that only authorized nodes have access to it. A typical example of confidential information would be a PIN code that is entered for accessing a room. However, data that is exchanged within an HVAC system may also contain confidential information. Consider a single room temperature sensor: while the current room temperature seems to be no secret, a low temperature may indicate that the HVAC system is in "vacation mode". Using this knowledge, an adversary may deduce that there is nobody present.
- *Data availability*. Data availability guarantees that authorized nodes have access to the data and that the access to this information is not prevented by adversaries.

Beside these primary security objectives, there are *secondary security objectives* that may be more or less relevant for building automation. *Anonymity* guarantees that an adversary cannot learn the identity of a node. Furthermore, it is guaranteed that an adversary is not able to track a node and that it is not able to derive a behavior pattern. While anonymity might be a side issue, it may be important when a node is directly related to an individual. A typical example would be a social alarm system that monitors the healthiness of people. *Auditability* is concerned with providing the proof what a system has done. It is only possible if several other security objectives like data origin authentication and non-reputability are guaranteed. Auditability is primarily important in systems like fire alarm or access control.

B. Quality of Service

Depending on the requirements of the control applications, necessary communication services must also guarantee different Quality-of-Service (QoS) properties related to security. The most important one is *reliability*. Services are said be reliable if the following properties can be guaranteed [5]:

- *Integrity*: Every message received was previously sent i.e., there is no corruption of a message while it is transmitted.
- *No duplicates*[2]: Every message is received at most once.
- *Liveness*: Every message is received at least once.

While integrity is required by most applications in building automation, preventing duplicates and guaranteeing liveness may be optional. For example, control applications that receive absolute values of control data (e.g., the present room temperature) do not care about duplicates since receiving the same value twice does not influence their proper functionality. If a control application receives absolute values at regular intervals, missing a few values may be tolerable and liveness may be optional. However, applications that send relative state changes (e.g., increase the set point of the room temperature by 5 degrees) demand a reliable communication service that guarantees liveness as well as the absence of duplicates. Otherwise, irregular system states may be the result.

In addition to reliability, the *ordering* of the messages may also be a requirement (see data freshness). In the general case of multiple sources and sinks, three kinds of ordering are distinguished:

- *Single-source FIFO ordering*: For all messages m_i, m_j and all nodes n_k, n_l, if n_k sends m_i before m_j then n_l does not receive m_j before m_i.
- *Causal ordering*: For all messages m_i, m_j and each node n_k, if m_i precedes m_j then n_k does not receive m_j before m_i where the "preceding operator" is defined by the well known "happened-before" relation [6].
- *Total ordering*: For all messages m_i, m_j and all nodes n_k, n_l, if n_k receives m_i before m_j then n_l does not receive m_j before m_i.

III. Secure Communication Architecture

To provide a generic solution that it is applicable to building automation systems of all sizes and types (including security-critical applications), the requirements identified in the previous section have to be satisfied. To achieve this, the proposed architecture is based on a modular, plugin-based, multi-protocol communication stack. Its main feature is the support for communication services that guarantee end-to-end security on a per-device level. The stack itself is partitioned into three layers (cf. Fig. 1). The *Network Specific Layer (NSL)* provides low-level communication services that are used to transmit messages over native network media. To be able to reuse an already existing network infrastructure, any data link/physical layer combination can be used. On top of the NSL, the so called *Security Abstraction Layer (SAL)* is located. The SAL is the key component within the stack. It abstracts the communication services of the NSL, enhances

[2]Note that these QoS properties are related to the control applications (i.e., to the application layer). At lower protocol layers, the requirements may be different. Consider, for instance, safety applications where message duplication is used to detect transmission failures.

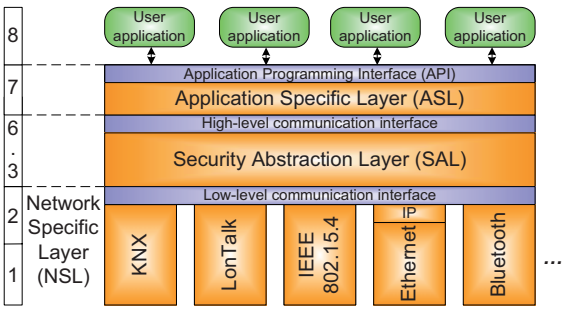

Fig. 1. Multi-protocol architecture

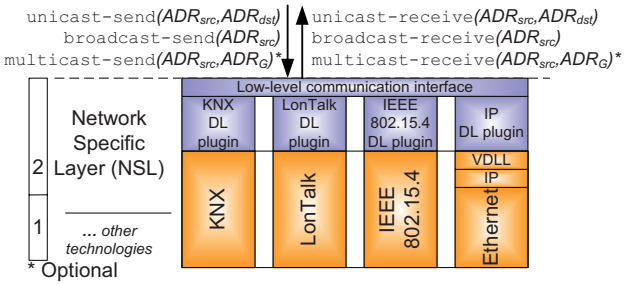

Fig. 2. Low-level communication interface between NSL and SAL

them with security, QoS, and routing/naming features, and offers generic secure communication services to the above located *Application Specific Layer (ASL)*. The ASL cares for the functionality of a common application layer and provides an interface to user applications. As all layers operate on plugins, easy extension is supported.

IV. Network Specific Layer

The NSL corresponds to layers 1 and 2 of the OSI reference model. It provides basic access to the underlying network medium and supports low-level communication services for sending and receiving messages on a single hop basis. Today, many different building automation technologies exist which in turn support various network media. Each of them offers significant advantages regarding their physical characteristics. While Ethernet based networks provide high bandwidth that is necessary for backbone networks, fieldbuses based on Twisted-Pair (TP) or Powerline are advantageous at the field level since they support free topology. Wireless technologies, with all their benefits and challenges, are also getting more and more important. Available technologies also differ at the data link layer. For example, some of them offer native support for multicast (e.g., KNX, LonTalk) which can be used to efficiently exchange data within communication groups.

As a result, choosing a single existing data link/physical layer combination or even developing a new one is not desirable. Therefore, the presented solution does not demand the use of a single data link/physical layer combination – in principle, it is possible to reuse any existing network technology. To provide an abstraction of the underlying data link communication services, so called *data link plugins* are introduced. A data link plugin is dedicated to a specific data link/physical layer combination and is located between the NSL and the SAL. Data link plugins are geared towards sensible (re-)use of already existing data link primitives. This means that each plugin chooses the services that fit best for the communication services required by the SAL. A typical example would be the reuse of the existing multicast communication services of KNX or LonTalk. In contrast to that, the use of unsuited protocol features (e.g., ineffective security mechanisms provided by KNX or LonTalk) can be blocked by the plugin. Furthermore, it is even possible that a device implements more than one data link/physical layer combination (multi-protocol devices). Therefore, devices are able to have several interfaces to heterogeneous networks.

In addition to native data link plugins, it shall also be possible to use a higher layer (i.e., layer 3 and above) as data link layer. To achieve this, a so called *Virtual Data Link Layer (VDLL)* is included. A VDLL simulates the use of higher protocol layers as native data link layer. A typical example would be the use of IP as data link layer for the SAL.[3]

To be able to use a broad range of network technologies, the demands on the underlying data link/physical layer combinations shall be reduced to a minimum. Therefore, only an unconfirmed unicast and broadcast communication service are considered as mandatory. Fig. 2 shows the resulting *low-level communication interface* that is located between the NSL and the SAL. The data link services `unicast-send` and `unicast-receive` are used to send and receive local unicast messages. ADR_{src} and ADR_{dst} denote the source and destination addresses used by the data link layer of the underlying network technology. To send and receive local broadcast messages i.e., messages to all network members located at the same network segment, the services `broadcast-send` and `broadcast-receive` are available. Other communication services (e.g., native support for multicast) or service features (e.g., confirmed communication) are optional. However, if they are provided by the underlying data link/physical layer combination, they may be reused by the corresponding data link plugins. A typical example would be use of services for multicast communication (cf. `multicast-send` and `multicast-receive` in Fig. 2).

V. Security Abstraction Layer

The SAL corresponds to the OSI layers 3 to 6 serving two objectives. First, it is responsible for providing a *routing* scheme that allows a communication across heterogeneous network segments. To achieve this, a global addressing scheme is introduced. This scheme is based on global *SAL addresses* that are used by the user applications to address other devices or groups of them (i.e., communication groups). Each device has to have a unique ID (denoted as ID_A) that acts as SAL address. In practice, this ID can be a serial number or a well-

[3]This concept is similar to BACnet/IP where UDP is used as data link layer for BACnet internetworks.

defined human-readable name[4]. Each network segment has a dedicated SAL network address N_x that is unique within the whole network. Between network segments, routers that implement the proposed communication stack are located. Since the network topology is static, it is assumed that routers are configured accordingly.[5] This means that each router knows the network addresses of its connected network segments and has sufficient routing information to find the next hop to any network segment. Using these SAL addresses, routing across network segment borders is possible – the SAL maps the node's ID (ID_A) to the network address where the device is connected to and also to its so called device address (denoted as ADR_A) and vice versa. A device address has to be unique within the network segment and is identical to the address used by the corresponding data link layer. Using this mapping scheme, the different data link layers can be abstracted and a heterogeneous routing is possible.

Additionally, devices can be arranged in communication groups. Since communication groups are not limited to a dedicated network segment, they are defined within the scope of the whole network. Each group has a dedicated group ID (denoted as ID_{G_x}) which acts as SAL address. Again, this ID is used by the application to uniquely identify a specific group. If the underlying network technology provides a native multicast service at the data link layer, ID_{G_x} is mapped to the group address(es) used by the data link layer(s) (denoted as ADR_{G_x}). Otherwise, the SAL maps the request to multiple unicasts or to a global broadcast.

The second objective of the SAL is to provide generic secure communication services to the ASL. The security concept of the SAL is based on the concept of *secure communication relationships*. A node can be member of one or more secure communication relationships. Depending on the amount of members, three different types of secure communication relationships are distinguished. A *network relationship* (denoted as N_x) consists of all members of a network segment. Relationships that contain only two members are referred to as *session relationships* or *sessions*. A session is denoted as S_{XY} where the nodes X and Y are the two members of the relationship. Finally, relationships that consist of three or more members that are located across the entire network are referred to as *communication groups*. A communication group is denoted as G_x where ID_{G_x} holds the unique ID within the network. Based on these three different types of communication relationship, all six communication types typically found in building automation [7] can be supported by the SAL: *secure point-to-point control data communication* and *device management* are performed within sessions using secure unicast, *secure loose group communication* and *network management* are handled by network relationships using secure broadcast, and *secure strict group communication* and *group management* are provided by communication groups using secure multicast.

To protect the communication services against security attacks, a secured channel is necessary. The basis for providing a secured channel is the use of cryptographic schemes. However, cryptographic schemes are computationally intensive. Since embedded devices with limited system resources (processing power, persistent and volatile memory, power consumption, and network bandwidth) are commonly used in building automation, the realization of a secured channel must not exceed the available device resources. Therefore, only those cryptographic schemes that are absolutely necessary to satisfy the security demands of the application shall be implemented ("good enough security"). For example, if the non-disclosure of the transmitted data is not a strict requirement, guaranteeing data origin authentication and freshness may be sufficient. Therefore, applications are able to choose between the following *security levels*:

- *Raw*: Raw communication services are not secured at all. In order to avoid an unauthorized manipulation of data, the communication channel must be secured physically.
- *Protected*: Protected communication services are services where only data integrity is guaranteed. Data freshness is not provided.
- *Trusted*: A communication service that is classified as trusted is secured in a way that data integrity and data freshness are provided.
- *Confidential*: Trusted communication services where the exchanged data is additionally encrypted are called confidential communication services. It is guaranteed that only authorized nodes are able to read the clear text version of confidential data.

For trusted and confidential communication services, guaranteeing data origin authentication instead of data integrity is optionally possible.

The communication services provided by the SAL are accessible through the so called *high-level communication interface* (cf. Fig. 3). To establish a session for secure point-to-point control data communication or device management, the `session-start` primitive has to be used. ID_{dst} denotes the ID of the destination node, $security$ the required security level (i.e., raw, protected, trusted or confidential with optional support for data origin authentication), $reliability$ the requirements regarding reliability (i.e., integrity, no duplicates, and liveness), and $ordering$ the desired ordering of the messages (i.e., single-source FIFO, causal, or total) demanded by the application. Message exchange during a session is done by the `session-send` and `session-receive` services. To terminate a session, the `session-end` service is available.

To join a network, the `network-join` service is present. Since it is possible that a node has more than one network interface, the parameter if specifies the interface where the join has to be performed. Again, $security$, $reliability$, and $ordering$ denote the requirements of the application. To exchange messages within network relationships, three different

[4]The distribution of the IDs is done during system deployment by the system engineer. Therefore, the IDs may be distributed randomly or according to a hierarchical assignment scheme based on the physical building structure (e.g., "buildingA/floor1/roomA/light1").

[5]Using a routing protocol that allows a change of routing information during runtime is possible but out of scope of this paper.

Fig. 3. High-level communication interface between SAL and ASL

kinds of services are available. The `network-send-local` and `network-receive-local` services are dedicated for local broadcast i.e., broadcast dedicated to members of the network segment where the device is connected to. The parameter if specifies the target interface of the local broadcast. The `network-send-remote` and `network-receive-remote` primitives are used to send and receive messages that need to be forwarded to a remote network segment. The address of the destination network segment is specified using the parameter N_{dst}. Finally, the `network-send-global` and `network-receive-global` services are available for addressing all members within the entire network. To leave the network where the device is connected to, the `network-leave` service is provided.

Using the `group-join` service, a device is able to participate in a group. ID_G specifies the ID of the group, $security$ the required security level, and $reliability$ as well as $ordering$ the QoS properties demanded by the application. Sending and receiving of group messages are possible using the `group-send` and `group-receive` services. To leave a communication group, the `group-leave` service is at hand.

To be able to fulfill the requirements regarding security, reliability, and ordering, the communication stack of the SAL is supported by the so called *security manager*. It is responsible for managing the membership of the different secure communication relationships (i.e., session establishment and termination, network/group join/leave). It has also the task to maintain the corresponding meta-data that are associated with the relationships. For each relationship, this includes the corresponding SAL and data link addresses, routing information to reach the specific members of the relationship, as well as the used cryptographic secrets to protect communication against security attacks.

VI. Application Specific Layer

The ASL corresponds to the application layer of the OSI reference model. It makes use of the communication services provided by the underlying high-level communication interface and offers an API to the hosted user application(s). The main aim of the ASL is to completely hide the complexity of the underlying communication system. For user application engineers, it shall be possible to focus on the implementation of the functionality of the desired control applications – that is the collection of input information, performing control functionality, and interacting with the environment by setting new output values. However, user application engineers should not need to bother with communication details. Managing communication relationships and dealing with data exchange between the user applications shall be left to the stack.

To provide such a high-level approach, the ASL is based on the concept of *data points* (cf. Fig. 4). They provide an abstract encapsulation of the control data that is under control of the user applications (e.g., the current output value of a light). The ASL is responsible for the management of these data points. All data points of a device are represented as so called *Application Objects (AOs)* stored in a generic *application object database*. A typical example would be a binary object of type "Boolean" that represents the output value i.e., the data point of a light.

To access these AOs, two different ways exist. First, the user application must be able to manipulate the AOs of interest. The access to the application object database shall only be possible through a well-defined API. Second, since control applications are typically of distributed nature, remote devices must also be able to access remote data points via the network. This is achieved by associating data points of one node with (multiple) data points hosted on remote nodes. These associations are also referred to as *bindings*. If two AOs located at two remote devices are bound with each other, changing the value of the data point at one side also changes the value of the corresponding data point at the remote side. Using this scheme, user applications can take full advantage of control data that is distributed across the entire network. Consider, for example, the AO of a light switch that is bound with the AOs of two lights (cf. Fig. 4). The necessary binding information is stored within a so called *binding table* that is under control of the ASL. Note that associations between AOs are not restricted to one-to-one relations. One-to-many or even many-to-many bindings may also be possible.

To be able to perform a reasonable binding between remote AOs, the structure and semantics of the associated data points have to be specified, too. This concerns the *data point type*, the corresponding *representation* (i.e., the encoding of the data points' values within network messages and their interpretation), and *meta-data* that is associated with the data point. Typical examples of meta-data among others are engineering units, upper and lower bounds, and most important the required *security* and *QoS level*. It must be possible to specify the minimum security and QoS level that the remote node must fulfill in order to be able to bind to a remote AO.

Beside the ability to change the value of data points, their management is also of great importance. This concerns creation, changing, and removal of AOs as well as their corresponding binding entries. These configuration and maintenance tasks shall be possible within two ways. First, the API shall provide user applications the opportunity to dynamically manage AOs and their associated bindings during runtime.

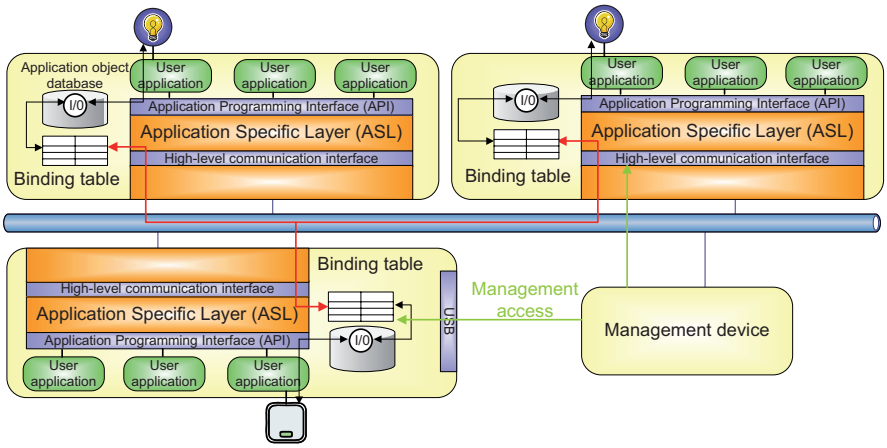

Fig. 4. Application model

Second, it shall also be possible to access the AOs using management tools. Management access can be provided via the network (using the same secure communication services as for user applications) or via a dedicated local interface (e.g., using a point-to-point connection). Obviously, to avoid malicious misuse, the management access must be protected.

VII. Conclusion and Outlook

This paper presented an adaptive security layer protocol architecture that is capable of operating on heterogeneous networks. The modular framework is designed to support virtually any combination of network protocols to meet the requirements for the integration of security-critical applications best. Through a plugin-based approach, easy extension and reuse of existing protocol mechanisms are achieved. When put into practice several non-functional requirements must be taken into account. The overhead imposed by the security mechanisms needs to be reasonably small. It is essential to find a good balance between a required level of security and available resources. For example, if the non-disclosure of the transmitted data is not strictly necessary, data confidentiality is unnecessary. Since building automation is usually home for a huge amount of devices, scalability of the integrated security mechanisms (e.g., secret key exchange and revocation) is of major concern. Moreover, building automation systems have to be kept operable for years or even decades. Since designing a perfect secure system is impossible, it must be assumed that security vulnerabilities will be discovered during the intended long lifetime. To be able to correct identified flaws in the system design, the possibility to update and maintain the used system components has to be provided.

To evaluate the presented architecture, a detailed specification as well as a proof-of-concept implementation are currently underway. Providing security objectives for the required secured channel is done using cryptographic techniques. Since cryptographic algorithms are computationally intensive, their use must not exceed the available device resources. Therefore, symmetric algorithms (e.g., HMAC for providing data integrity, AES for en/decryption of confidential data) as well as asymmetric ones that are suitable for embedded devices (e.g., algorithms based on elliptic curve cryptography) have to be used. Integrating security mechanisms also assist in satisfying reliability properties. If the used cryptographic algorithm already provides data integrity, integrity (from a reliability's point of view) is guaranteed in a native way. Mechanisms to guarantee data freshness can also be reused for reliability and ordering issues. For example, using monotonically increasing counters provide data freshness but also work against duplicates. If each sender has its own counter even single-source FIFO ordering can be guaranteed. If more sophisticated time variant parameters are used for data freshness, even causal or total ordering can be supported. For instance, both, logical vector or synchronized timestamps guarantee data freshness, and offer causal and total ordering, respectively. However, some QoS properties cannot be supported by using security mechanisms exclusively. To provide liveness, additional measures based on feedback of the receiver(s) have to be implemented. Since a simple acknowledgment mechanisms may lead to inconsistent data views (especially in communication groups), more advanced schemes are necessary. Here (two/three) commit phase protocols may be a solution.

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, Jun. 2005.

[2] W. Kastner and T. Novak, "Functional safety in building automation," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation*, Sep. 2009.

[3] W. Granzer, F. Praus, and W. Kastner, "Security in building automation systems," *IEEE Transactions on Industrial Electronics*, vol. 56, 2009.

[4] R. Shirey, "Internet Security Glossary, Version 2," RFC 4949, August 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4949.txt

[5] H. Attiya and J. Welch, *Distributed Computing—Fundamentals, Simulations and Advanced Topics*. Wiley-Interscience, 2004.

[6] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[7] W. Granzer, C. Reinisch, and W. Kastner, "Key set management in networked building automation systems using multiple key servers," in *Proc. IEEE International Workshop on Factory Communication Systems*, May 2008, pp. 205–214.