HomePort ZigBee Adapter

Internal Milestone Report

Pedersen, Thomas; Smedegaard, Jacob Haubach; Hansen, Rene

Publication date:
2014

Document Version
Accepted author manuscript, peer reviewed version

HomePort ZigBee Adapter

(Internal Milestone Report)

Thomas Pedersen*, Jacob Haubach Smedegård* and Rene Hansen*
* Department of Computer Science
Aalborg University, Denmark
Email: {*tp,jhaubach,rhansen*}@cs.aau.dk

August 18, 2014

I. Introduction

The ZigBee protocol is a large and complicated standard with multiple abstraction layers in both hardware and software. Based on the IEEE 802.15.4 physical network layer, the ZigBee Alliance platform extends it with a network and security layer, application support sub layer, application profiles and finally the actual application itself. Ultimately, a ZigBee implementation is a substantial undertaking for new players in the field, or companies with already established products and proprietary communication stacks. The purpose of this project is to enable a non-ZigBee device, such as the ConLAN keypad to access, and be accessed from, a ZigBee device network.

To accomplish this we utilise the existing tool, Homeport, to act as a middleware and bridge between ConLAN existing network and the ZigBee network. Homeport [1] provides a RESTful interface to a diversity of different home automation networks and protocols, including ConLAN's. It can easily be extended with further adapters to support additional networks. Thus this report considers the addition of a ZigBee Adapter, using a gateway provided by Develco.

A. ConLAN

ConLAN[1] is a small danish company producing keypads and various readers for access control. Their products can be connected in a network and integrated into other systems.

B. Develco Products

Develco Products[2] is a development company with electronics and embedded systems. It this project they are the experts on the ZigBee platform and provides the necessary hardware to interface with it.

II. Simplified ZigBee Profiles

Enabling interoperability between vendors and heterogenious devices is possible due to the definition of Profiles and Clusters. Profiles determine the application domain such as Home Automation, Smart Energy, Health Care and Green Power, whereas Clusters define groupings of related functions. Each cluster defines a set of mandatory functions that a device supporting this cluster must implement, and a set of functions that may be optionally supported. Since the ZigBee defined profiles are comprehensive and the accompanying ZigBee Cluster Library even more so, we have defined proper subsets of these profiles to enable basic functions for new devices. Thus they are compatible with the full profiles.

The simplified ZigBee Profiles are based on the ZigBee Home Automation Profile and ZigBee Cluster Library (ZCL) specification.

The developed profiles fall into the following five categories:

- General
- Lighting
- Closures
- Heating Ventilation Aircondition (HVAC)
- Intruder Alarm System (IAS)

The General profile contain the minimum set of attributes for all devices, such as the ability to turn a device on/off or control its level, whereas the remaining categories add domain-specific attributes. The attributes for the different profiles are the mandatory attributes from the ZCL specification. Also, we have chosen to adopt the ZCL approach of grouping logically coherent attributes into clusters that can be used by different profiles. We refer to the file `devices.c` for details concerning the introduced profiles and their attributes.

III. HomePort Adapter

To realise the simplified ZigBee profiles, we have to implement a ZigBee adapter in HomePort. This adapter will utilise the profiles to facilitate a two-way communication between devices on HomePort and devices on ZigBee. The aim is for HomePort to act as a bridge between the non-ZigBee devices connected to HomePort and the ZigBee network.

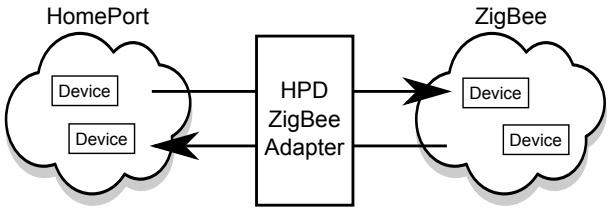

Fig. 1. Purpose of the project

[1] ConLAN's website: http://www.conlan.eu
[2] Develco Products's website: http://develco.dk

As a regular HomePort adapter, it will expose all ZigBee devices as RESTful webservices in HomePort, and thus the ZigBee devices can be accessed and controlled like any other HomePort device. In contrast to traditional HomePort adapters, the ZigBee adapter facilitates visibility in the ZigBee network. That is, any device already on the HomePort network, e.g. the ConLAN keypad, may be visible to the ZigBee network as well. Ideally, this visibility should be transparent, such that a ZigBee device can discover the keypad and communicate with it, as it were a regular ZigBee device. Section VI discusses several possible solution for obtaining more or less transparency.

IV. Preliminaries

This section gives a brief introduction to ZigBee for uninitiated readers. Those that want to go into deeper details, please consult the ZigBee website and specifications.

A. The ZigBee Network Structure

In a ZigBee network each device can have one of three roles; coordinator, router and (end-)device. The hierarchy is shown in Fig. 2. There can at most be one coordinator, but there may be any number of routers and devices. In the case that a coordinator fails, a router may take over this particular role.

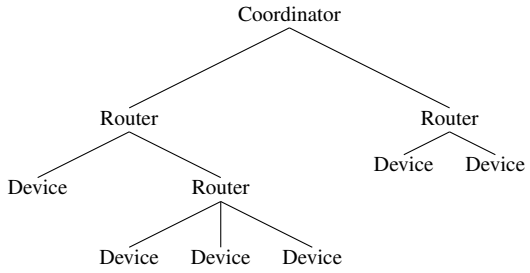

Fig. 2. Roles in a ZigBee network

1) Coordinator: assigns network addresses to devices through authorization. The coordinator also acts as a router in the network.

2) Router: routes messages between devices in the network. As routers are used to extend the range of the network, a message may pass through any number of routers.

3) End device: is a simple device on the network (e.g. sensor or wall socket).

B. Addressing on a ZigBee Network

The addressing on a ZigBee network looks very similar to that of a regular TCP/IP network. Each ZigBee device have a global unique EUI-64 address (think of a MAC-address), a 2-byte network address (think of the IP address) and a 1byte endpoint address (think of a port). Devices can be addressed either by its EUI address or by its network address. Like the traditional TCP/IP network the network address may change over time, especially if the device is disconnected from the network and later reconnected.

The approx. 256 possible endpoints on each device is used to identify different applications. For example it is common to find ZDO (ZigBee device objects) on endpoint 0x00, which handles information about known devices in the network.

V. Develco Products' USB Gateway

The USB Gateway, kindly supplied by Develco Products, is an intelligent access point to the ZigBee network. This means that it is capable of handling the lower layers of the ZigBee network, and thus we do not have to care about every single detail. However, it also means that when communicating with the ZigBee network we have to communicate with the gateway as well. For this the gateway comes with its own protocol.

The messages are structured in different layers, each with different headers and footers. On the lowest level the USB gateway registers as a TTY device, usually on `/dev/ttyACM0`. All messages are just a stream of bytes in UART datagrams. UART specifies that the first byte of a message is 0x02, the second is the length and the last is 0x03. The intermediate bytes are interpreted as SmartAMM messages - A Develco specific layer, that is used to communicate with the different modules on the gateway itself. An example of a module is the gateway application (GwApp) that can give status messages about the gateway, for example about memory usage. Such a message is shown in Fig. 3.

The most important module in this project is the modem module. It handles all messages that are sent to and from the ZigBee network itself. Each of these messages have different types and addresses that are used to determine how to interpret their contents. The contents is pure ZigBee messages as described by the ZigBee protocols. An example is shown in Fig. 4.

VI. ZigBee Extension Solutions

Allowing non-ZigBee devices to participate in the ZigBee network as ZigBee-enabled devices with bindings on input and output clusters requires some virtualisation of the device on the Gateway. Possible solutions are 1) create a virtual ZigBee router on the ZigBee network to route messages to non-ZigBee devices, 2) the Gateway act as a blackbox device comprised of the hardware of all HomePort connected devices or 3) let devices bind directly to a single backend endpoint in the Gateway.

A. Solution 1: Fake ZigBee Router

In the fake ZigBee router solution, HomePort (HP) presents itself as a fake router to the ZigBee network, see Fig. 5. Homeport will then announce its devices as ZigBee devices routed through it. As HP acts as a router on the network, any ZigBee device will try to route messages through it in order to reach the HP devices that are connected to it.

What they do not realise is that HomePort itself is answering on these message. Consider when a new non-ZigBee device is connected to HomePort. In this scenario HomePort will generate and transmit a message onto the ZigBee network, telling the network about the new device. This message will

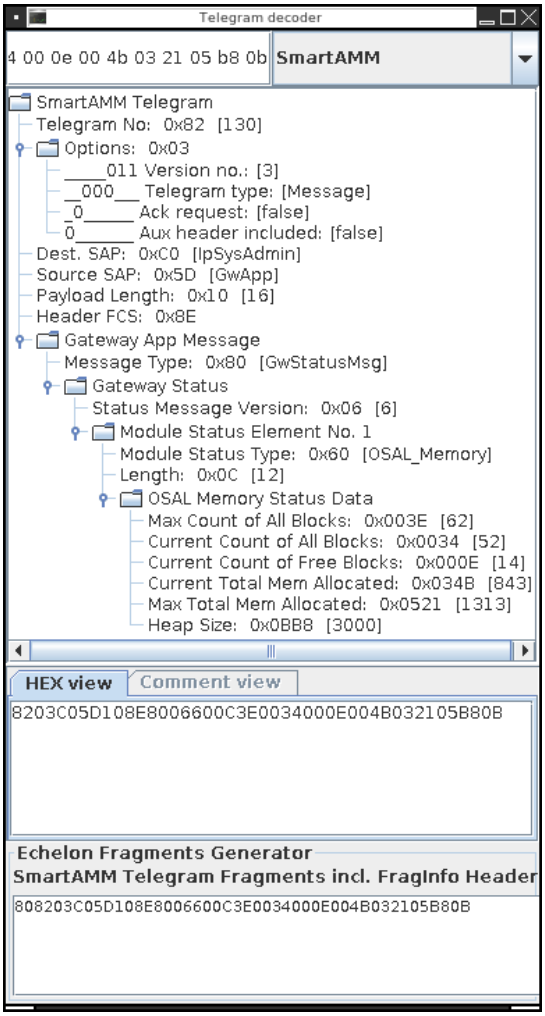

Fig. 3. Example of a message from the GwApp module, decoded by Develco's SmartAMM Developer Utility

be generated by HomePort to present itself as if it originated from the device itself. The ZigBee network will then accept this device in the network and assign a network address to it. An address that will be stored in an internal mapping table in HomePort between the virtual ZigBee device and the real Non-ZigBee device.

Later if the ZigBee network wishes to transmit a message, such as a read attribute request, to the device, this request will be routed according to the routing information to the HP router to reach the corresponding device. However, HomePort does not relay the message, but parses and interpret it itself. To answer the message, HomePort may first have to communicate with the device over the non-ZigBee network. Hereafter, HomePort replies to the message on the ZigBee network again making the message look as if it came from the virtual ZigBee device, routed through HP Router in the network.

Currently, the product lineup from Develco Products, does not support this type of operation. Thus, it is currently an infeasible solution to the problem.

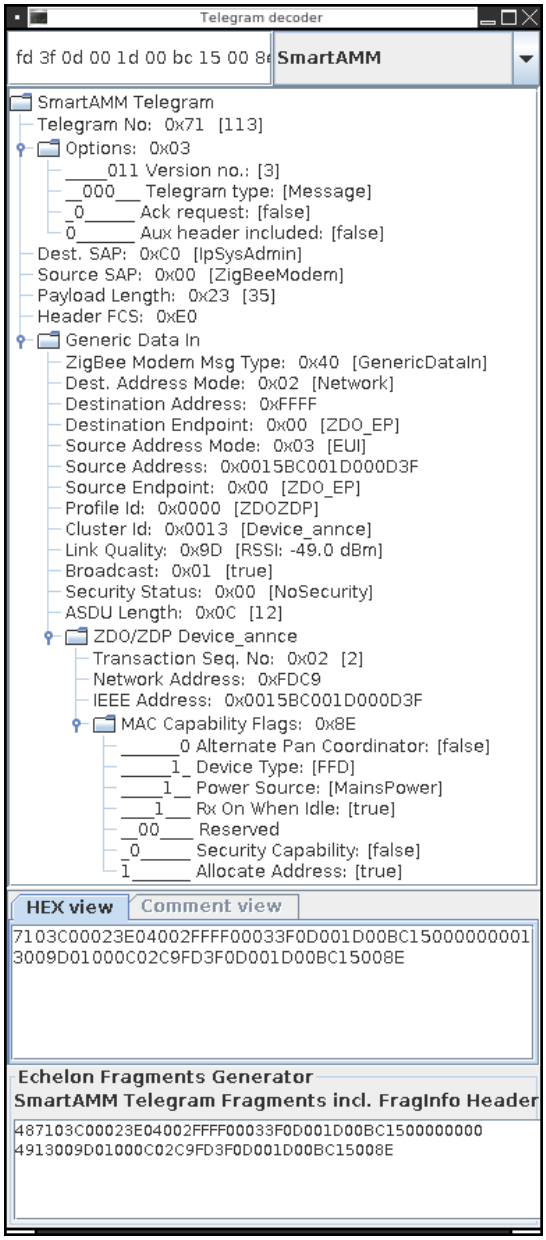

Fig. 4. Example of a message from the modem module, decoded by Develco's SmartAMM Developer Utility

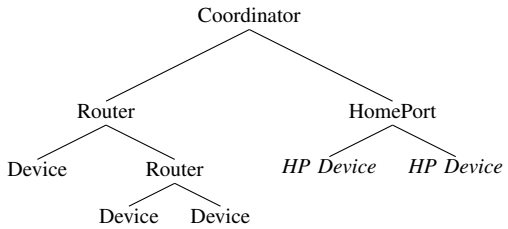

Fig. 5. Solution 1

B. Solution 2: Endpoints

The USB gateway currently provided by Develco Products have been designed to run in Coordinator mode only, thus we

shall assume this role for HomePort. However, do note that this solution could also be implemented as other roles.

Usually each device does not utilise every possible endpoint, but only a handful of these. It is therefore theoretically possible to connect each HomePort device to their own endpoint(s) on the HomePort coordinator. In this solution, ZigBee devices will see HomePort as one single device on the network the coordinator. However, when scanned for its capabilities, it will announce that it will have the hardware capabilities of a wall socket on endpoint 0x10, a keypad on 0x11, etc.

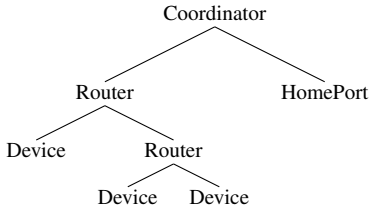

Fig. 6. Solution 2

This solution have two important limitations. First, the devices on HomePort are not seen as real ZigBee devices, but as a single device. This may cause problem during device discovery. Secondly, the number of endpoints is limited to approx. 256, thus the number of devices that can be supported by HomePort will be limited likewise and there will be limited segregation between actual devices.

The current USB gateway is based on a Texas Instrument chip that does not support the creation of additional endpoints. Develco Products is currently working on a Freescale USB gateway that will solve this issue. Until then, we can only provide a single device through the already active endpoint on the USB gateway.

C. Solution 3: Non-transparency

A final option is to let HomePort be HomePort in the ZigBee network and avoid any transparency in the solution that would allow ZigBee devices to perform discovery as usually. One method would be to design an interface to the HomePort webservice. In this solution, ZigBee devices need to be aware of commands, addressing schemes etc. in HomePort. Therefore it will not be very generic and will not solve the problem of getting a non-ZigBee device on a ZigBee network.

D. Current Target Solution

As Solution 1 is currently infeasible and Solution 3 is violating the desire for a transparent solution, the current target is Solution 2. The plan is to produce a working prototype on the single endpoint available now and then scale it to multiple endpoints at the time Develco Products delivers a Freescale USB gateway. The ZigBee messages involved in all three solutions are covered by the same ZigBee protocols. Thus, it is expected that the interpreter for Solution 2 can be reused for Solution 1, if a solution supporting more devices is needed.

VII. Adapter Architecture

To adhere to the message structure and allow as much flexibility as possible to add additional commands, the adapter has been designed using layers that correspond to the packaging of the messages, see Fig. 7.

A. TTY layer

Is used to open the TTY connection and read/write the byte streams. This is shared with other HomePort adapters that opens their connection through TTYs.

B. UART

The UART layer chops the bytestreams into messages by looking for the start and end bytes of the messages and verifying that the length matches. If the length does not match, we assume that we are in the middle of a partially received message and skips ahead until the next start byte.

C. Develco Layer (SmartAMM, Modem, Develco)

SmartAMM parses the SmartAMM headers of the UART payloads. The header may specify different types of the message. Some of these messages can be handled directly in the SmartAMM module. This includes gateway ids (announcements of the EUI-64 address of the gateway), time requests from the gateway itself, and some ACK handling. However, the more complex type of messages, the *message* type, specifies a payload that has to be parsed. The type of payload depends on a SAP address.

Erroneous messages are detected by mismatch in length, check sequence, or data. These are handled by skipping to the next UART start byte, thereby considering that they may have been a later part of a partially received message that just happens to match correctly in the few first bytes.

Current only messages from the modem are interesting, however other handlers can easily be registered in SmartAMM. The Modem module parses these messages.

The Develco Module collects a single interface to the Develco layer and handles the start of the SmartAMM and Modem modules.

D. ZigBee Layer (ZCL, ZDO, ZigBee)

The ZigBee endpoint `0x00` is designated to handle ZigBee Device Objects (ZDO). For example this endpoint receives announcements of new devices on the network and requests for devices that support a given cluster. The ZDO module maintains a list of known ZigBee devices and queries the ZCL module for information about registered clusters.

Commands specific for a given cluster are sent on to the other endpoints and will be handled by the ZigBee Cluster Library (ZCL) module. This module maintains a list of registered endpoints, profiles, clusters and attributes in the adapter. Each endpoint can support one or more profiles, under each profile one or more clusters can be registered, and each cluster has one or more attributes. Attributes have associated function pointers

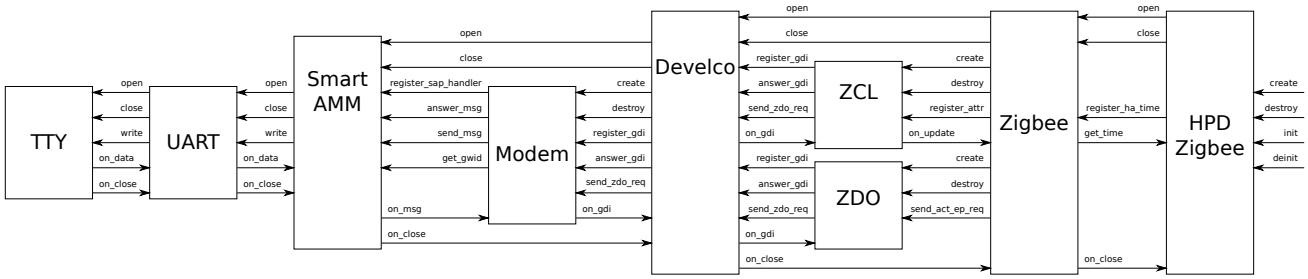

Fig. 7. Adapter Architecture

that are used when read and write requests are received for a given attribute.

The ZigBee module is, like the Develco, a single interface to the ZigBee layer and handles the start of the ZCL and the ZDO modules, as well as the Develco layer.

E. HPD ZigBee

The HPD ZigBee module is the HomePort adapter itself and provides a simple interface to create and initialise it, exactly as every other HomePort adapter does.

VIII. Handled Messages

The descriptions and diagrams below reflect the current state of the adapter. There will therefore be examples of unanswered messages, that are currently not handled.

A. SmartAMM - Gateway IDs

At start up, the adapter sends a request for the ID of the gateway, see sequence chart in Fig. 8. The ID is required to ensure that messages are addressed to us and therefore no other messages are handled by SmartAMM until the ID has been received. As the architecture shows (Fig. 7), the messages are sent thought TTY and UART (not shown in sequence chart).

SmartAMM will automatically respond with ACK for all messages that requires it. The ACK response will not be shown for the other message types.

Example of messages in hex, without header and footer of UART diagrams[3]:

```
Gateway ID req: 00 13 5d c0 00 8e
Gateway ID:     08 53 c4 5d 08 c2 58 00
                00 21 00 bc 15 00
ACK:            08 0b 5d c0 00 96
```

B. SmartAMM - Time requests

To keep time synchronisation, the gateway regularly (every minute) requests the time. A request we kindly respond to with a UNIX timestamp with epoch at 1970, as shown in Fig. 9.

Time synchronisation example:

[3]Messages are displayed in a format that always copying them to the decoder utility by Develco Products

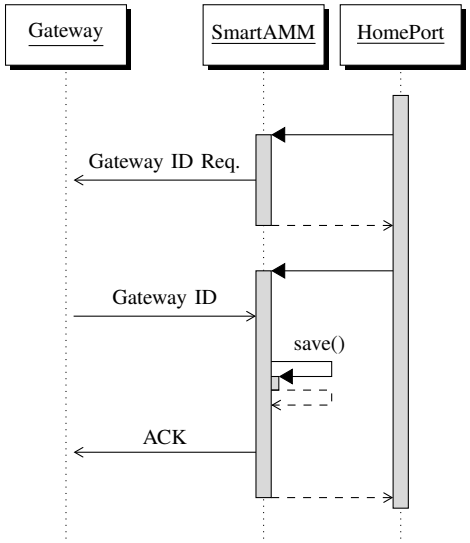

Fig. 8. Gateway ID request and response

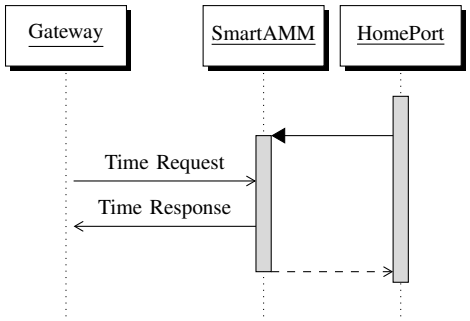

Fig. 9. Time request and response from gateway

```
Request:  0c 1b c4 40 00 9f
Response: 0c 23 40 c0 05 a6 00 f5
          91 74 53
```

C. Gateway App - Status Messages

The Gateway App regularly sends status messages about free memory, etc. These messages are currently ignored in

SmartAMM, as there are no registered handlers for message from the SAP address of the Gateway App, see Fig. 10.

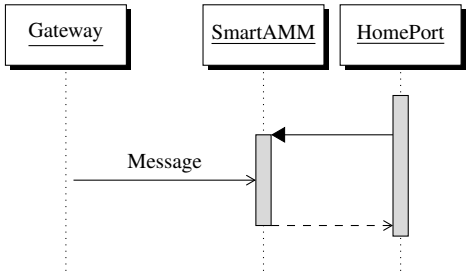

Fig. 10. Status messages from GwApp

Examples of messages:

```
Message: 81 03 c0 5d 2a b4 80 06
         01 26 ff ff ff 02 04 02
         00 00 00 02 05 01 01 00
         00 34 d8 11 1b 68 10 00
         00 ff ff 02 bb 04 01 00
         0a 00 00 00 00 00 00 00
Message: 82 03 c0 5d 10 8e 80 06
         60 0c 3e 00 34 00 0e 00
         4b 03 21 05 b8 0b
```

Current console output on these messages:

```
[smartamm] No registered handler for
    source 0x5d [GwApp]
```

D. ZigBee Modem - Device join

Several messages are transmitted when a device (re)joins the Zigbee network, shown in Fig. 11. This concrete example is with a Develco ZHWR202 (ZigBee Wall Plug Meter Relay HA) device. Note that time requests from devices are handled seperately (see below), but are also transmitted as part of join sequence. All messages are received/sent through the SmartAMM and underlying layers (not shown).

The first message is from the TrustCenter telling us that a device has rejoined the network. These message types are currently being ignore in ZigBeeModem. The second message is a request for device that can provide the time, an request that we respond with ourselves as a possible provider. The third and final message is a broadcast announcement of the device itself. We register its network address in a table of know devices. Both the second and third message are message for the ZDO.

Example of messages:

```
TrustCenter:   6d 03 c0 00 14 d7 42 58
               00 00 21 00 bc 15 00 3f
               0d 00 1d 00 bc 15 00 c9
               fd 03
MatchDescReq:  6e 03 c0 00 20 e3 40 02
               00 00 00 03 3f 0d 00 1d
               00 bc 15 00 00 00 00 06
               00 9d 00 00 09 43 00 00
```

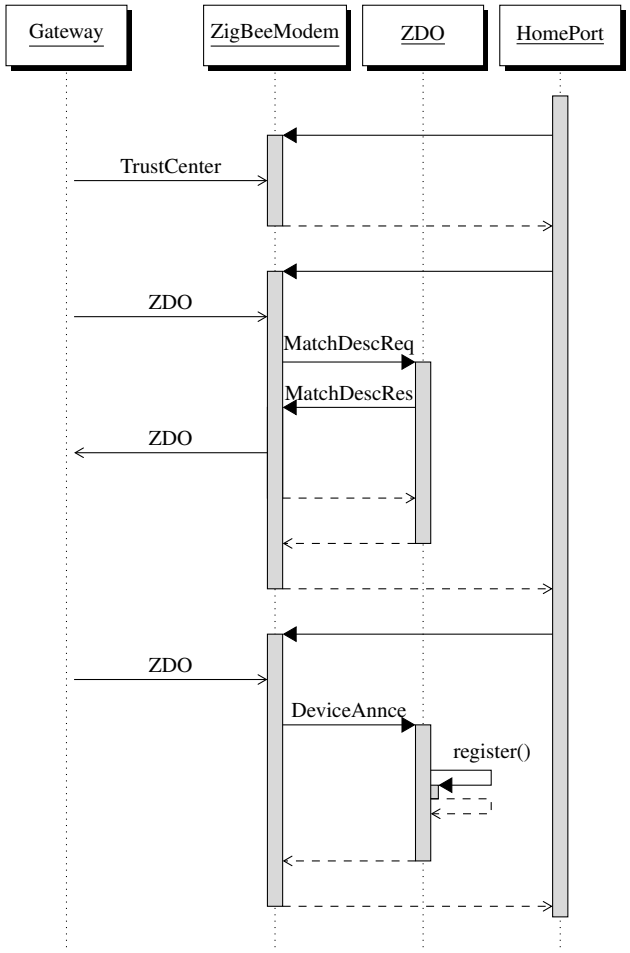

Fig. 11. Device joins

```
               04 01 01 0a 00 00
MatchDescRes:  6e 03 00 c0 18 db 00 03
               3f 0d 00 1d 00 bc 15 00
               00 00 00 06 80 00 00 06
               43 00 00 00 01 20
DeviceAnnce:   71 03 c0 00 23 e0 40 02
               ff ff 00 03 3f 0d 00 1d
               00 bc 15 00 00 00 00 13
               00 9d 01 00 0c 02 c9 fd
               3f 0d 00 1d 00 bc 15 00
               8e
```

Console output for unhandled messages:

```
[modem] Cannot handle
    TrustCenterUpdateDevice messages yet
```

E. ZigBee Modem - Read time attribute

As the gateway, a device may regularly requests the time, see Fig. 12. These messages differs from the previous mentioned time requests from the gateway. They are packaged differently, as a read attribute request within the ZigBee specification.

The first of the three messages in the sequence is the read attribute request. This is answered with the correct timestamp, with epoch in 2000[4]. The other two, a confirm and a success message, are not yet handled.

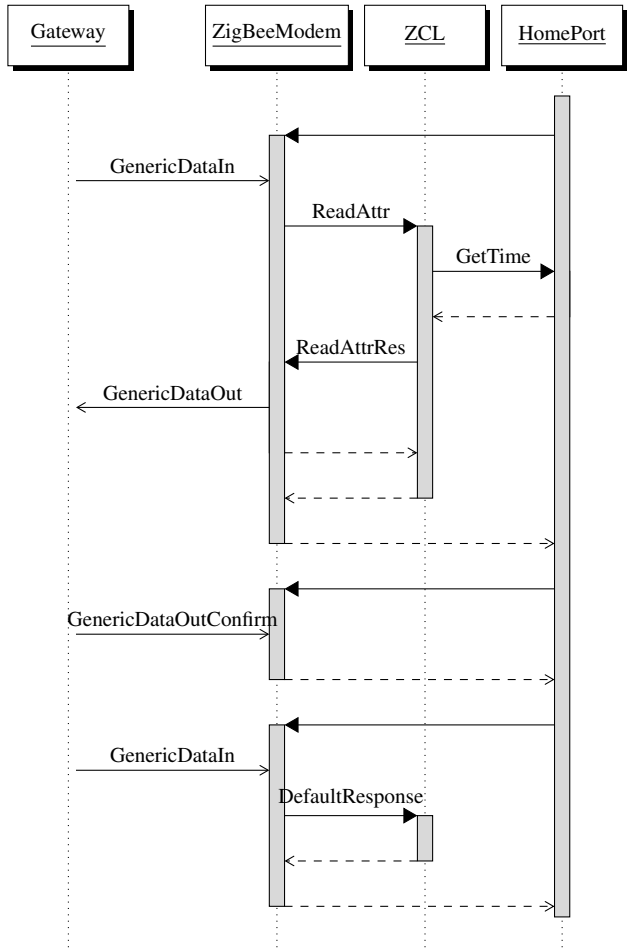

Fig. 12. Read attribute

Examples of messages:

```
Request:  7e 03 c0 00 1c df 40 02
          00 00 20 03 26 03 00 1d
          00 bc 15 00 21 04 01 0a
          00 93 00 00 05 00 42 00
          00 00
Response: 7e 03 00 c0 1d de 00 03
          26 03 00 1d 00 bc 15 00
          04 01 21 0a 00 ff 00 0b
          08 42 01 00 00 00 e2 1c
          86 07 1b
Confirm:  7f 03 c0 00 12 d1 80 03
          26 03 00 1d 00 bc 15 00
          21 c0 00 0c 78 07 1b 1b
Default:  80 03 c0 00 1c df 40 02
          00 00 c0 03 26 03 00 1d
          00 bc 15 00 21 04 01 0a
          00 93 00 00 05 18 42 0b
          01 00
```

Console output for unhandled messages:

```
[modem] Cannot handle
    GenericDataOutConfirm messages yet
[zcl] Cannot handle requests with disabled
     default response yet
```

IX. Conclusion

The overall purpose of the project is to ease access to ZigBee for new players or companies with already established products or communication stacks. To accomplish this we suggest to utilise the existing tool HomePort to bridge ZigBee with already existing networks, exemplified by ConLAN's keypads and access control. From the perspective of ConLAN, the solution will enable them to integrate their existing network into a ZigBee network by performing two tasks. First, they need a working adapter for including ConLAN devices in HomePort[5]. Thereafter, they need to provide enough information about their products to generate the simplified ZigBee profiles. Exactly how and in which format to provide this is still to be investigated. However, it is expected that this information is to be included within the ConLAN adapter itself.

The main concern in this report have been to investigate the feasibility of producing a ZigBee adapter for HomePort. This adapter should connect to the ZigBee network and, by the use of the information from the ConLAN adapter, expose the ConLAN devices. Thereby every ZigBee device can connect and communicate with the ConLAN device, as it were a ZigBee enabled device itself. The result of this approach is that we do the heavy work of understanding and adhering to the ZigBee specifications and ConLAN only needs to understand the simplified profiles.

Technically, we found three possible solutions for connecting HomePort to the ZigBee network. Unfortunately current available products does not support the first, which would have allowed the greatest flexibility. The chosen Solution 2 was theoretically limited to a maximum of 256 devices (a little less as some endpoints are already taken by the gateway). However until a new USB gateway is produced, we are stuck with a single endpoint. The third solution did not achieve the transparency we strove for.

Using the USB gateway, we successfully connected the adapter to the ZigBee network, including reception and reply of the basis messages involved in this process. Parsing messages and building responses clearly showed why companies find it troublesome to develop ZigBee products. The messages are layered within layer and each layer has types or addresses used to determine how it is to be parsed. This is reflected in the architecture, that has been designed to easily be expanded as further message types needs to be included in the parser.

[4]Zigbee uses different epoch (2000) than the USB gateway (1970)

[5]A ConLAN adapter for HomePort has already been produced by previous projects and is available on the internal HomePort SVN.

Currently no work has been accomplished towards the actually bridging of device, but parsing device announcements and replying to read time attribute messages from the ZigBee network.

Expect for the limitations in available hardware, discussed with the three possible technical solutions. A two-way communicating ZigBee adapter for HomePort does indeed seem feasible, however it may require a greater amount of time to produce. It has to be noted that we only considered hardware supplied by Develco Products in this project and the needed hardware for both Solution 1 and 2 are in the planning.

The current state of the software is available from the internal HomePort Subversion repository under `HomePort-Adapters/hpd_zigbee/` as source code. Neither source nor executable is public available. It have to be determined if the ZigBee and Develco specific details in the code can be publicly released and whether licenses for these is to be included.

It has been developed as a HomePort adapter in C and thus require prior compilation and installation of HomePort[6]. It has been developed and tested on Ubuntu Linux. Besides the requirements of HomePort the adapter shares some components with other HomePort adapters (e.g. a TTY component that the ConLAN adapter also utilises). These have been directly linked into the source and thus does not require any additional installation.

ACKNOWLEDGMENT

The work presented in this report has been supported by InfinIT, a Danish network for innovative utilisation of IT.

REFERENCES

[1] T. Le Guilly, P. Olsen, A. P. Ravn, J. B. Rosenkilde, and A. Skou, "Homeport: Middleware for heterogeneous home automation networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 627–633.

[6]The source from HomePort is publicly available from https://github.com/home-port/HomePort