The HomePort System

Brønsted, Jeppe; Madsen, Per Printz; Skou, Arne; Torbensen, Rune Sonnich

Published in:
2010 7th IEEE Consumer Communications and Networking Conference (CCNC)

DOI (link to publication from Publisher):
10.1109/CCNC.2010.5421606

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

The HomePort System

Jeppe Brønsted
Aarhus University
Computer Science
Aabogade 34
DK-8200 Aarhus N, Denmark
Email: jrb@cs.au.dk

Per Printz Madsen
Inst. 8 - Procescontrol
Fredrik Bajers Vej 7
DK-9220 Aalborg East
Email: ppm@es.aau.dk

Arne Skou
Aalborg University
Department of Computer Science
Selma Lagerloefs Vej 300,
DK-9220 Aalborg East, Denmark
Email: ask@cs.aau.dk

Rune Torbesen
Aalborg University
Department of Computer Science
Selma Lagerloefs Vej 300,
DK-9220 Aalborg East, Denmark
Email: rto@cfsi.dk

***Abstract*—Residential gateways for home automation are prerequisites to obtain optimal exploitation of energy resources, and they also have the potential to provide a unified operation of various home devices and appliances. Although a number of protocol standards have been proposed, the number of commercially available systems are still very limited. One reason for this is the diversity of device manufacturing standards, another is the lack of efficient and expressive middleware for defining control algorithms and usage scenarios. In this paper we present the architectural design of a distributed middleware system for residential gateways including a simple composition language. Also, we present the initial experiences obtained from a prototype implementation of the system.**

I. Introduction

In the last couple of years computer based home control systems are getting more and more common in modern homes. The main reason for introducing computer based control systems, in our homes, is to increase the comfort and security as well as to lower the consumption of resources such as energy and water.

However, the penetration of home control technology in the general population is still very low, due to a number of obstacles: high initial cost, incompatible protocols and lacking interoperability between systems [1].

Today a large number of home control systems are available on the market. Each of them takes care of one specific task for instance controlling of the heat or the light. These dedicated control systems (Subsystems) are often not capable of communicating with each other, because they use their own specific communication method mostly based on standard low-level wireless protocols. There exists no single dominating wireless protocol for home control systems. Currently, the main contenders for the market are Z-Wave [2] and ZigBee [3]. While there is a significant mass of z-wave products on the market today, Zigbee is only slowly gainig momentum.

The main problem is: If you want to increase both the comfort and the security and at the same time lower the consumption of resources you must have a system, which can interconnect the different subsystems in a flexible and intelligent manner. For instance, when you leave the house and lock the door, all the light should be switched off and the heating system should settle on a lower level and maybe the ventilation system should ventilate the house if the humidity is too high, etc. Of course all these control actions shall be executed in a system, which can communicate with the door lock (the security system), the light, the heat and the ventilation system. In addition to that, it is clear that the control actions depend on the state of the environment, the system itself and the people living in the house.

Besides achieving the goal of connecting home devices from different subsystems to achieve improved comfort, security and optimized energy consumption, the architecture should also fulfill a set of business goals as well as exhibit a set of architectural qualities (as defined in [4]).

Concerning the business goals, a requirement for achieving wide adoption is that it is profitable for businesses to use the architecture. Previously, a barrier to the integration of home devices has been that businesses have been reluctant to adopt open protocols. We conjecture that this is partly because it has been unclear to businesses whether a sustainable business model based on open protocols can be maintained. Therefore, it is a requirement that the architecture, to a high degree, supports current traditional business models based on partly closed protocols. Additionally, the infrastructure should have no impact on existing end-product designs and should also allow updating of these designs.

To achieve wide adoption, it should be possible for all interested parties to join the HomePort system and thus no single commercial authority should be in control of the system as a whole and in particular of the protocols used.

In addition to providing increased comfort and security, the energy saved should be a motivating factor for the consumer to adopt the system. To achieve this, the cost of the infrastructure should be low enough in order not dominate the cost of the energy saved. This implies that the nodes in the system should be embedded controllers and not, e.g., standard PCs.

The business goals are complemented with requirements for architectural qualities:

1) Modifiability: Once deployed, it should be possible to add new devices and subsystems to the system without affecting already deployed equipment. Furthermore, it should be possible to connect new added devices to already existing applications as well as new ones.

2) Usability: A significant part of the cost of current smart home systems is the cost for installation and configuration by technicians. If this task can be done by the user instead, significant savings can be achieved. This requires, however,

that the user is able understand and configure the system.

3) Scalability: Since the current trend goes towards more and more electronic devices in the home, the architecture should scale to include hundreds of devices, maybe even spread over a large building, without incurring a performance penalty on individual devices. Simultaneously, the architecture should scale to low end devices to ensure cost efficiency.

To achieve wide adoption of home automation with benefits of increased comfort, security and energy conservation, this paper proposes a communication infrastructure for connecting these different subsystems and devices. The infrastructure is based on service-oriented architecture (SOA) and introduces a number of well defined protocols and interfaces to allow market actors to produce homeport compatible components. Control logic is located in a generic composition layer that is isolated from subsystem specific protocols.

The rest of the paper is structured as follows. Below, we describe related work with respect to technology integration in home automation. In section III, we describe the Homeport system architecture. Following that, in section IV, we present a language for expressing service composites in the Homeport system. Section V presents an experimental setup, and section VI summarizes conclusions and outlines future work.

II. Related Work

To obtain interoperability in home automation between multiple standards [5] uses a central server to translate between end-devices. Their focus is on software based service development and they introduce a technology abstract service language (DomoML), that makes it easier to write generic services i.e. graphical user interfaces that via the server-gateway communicate with end devices. Once installed, this server will not be able to allow new end-device types (subsystems) to be added over time, since it requires updating DomoML language translation code in the server. Compared to our approach, our focus is on making an IP protocol/framework that makes it easy to integrate any wireless home automation technology into the system and write controller/user interface/composition software. Regarding reliability, our design does not require a particular deployment and many deployments are possible using one or more gateways/bridges. If required, a decentralized protocol translation can be obtained, allowing the translation to be distributed into sub IP networks. Also, our system is supporting existing equipment as subsystems. When translation is distributed, IP language updates requires only installing an extra translator gateway that is an embedded device available from any infrastructure device vendor.

With focus on middleware for services on a dedicated server (OSGI), [6] handles interoperability using software bridges on the server in order to integrate heterogeneous networks and devices. The solution suffers from the single point of failure problem and does not scale down to a small installation in a cost effective way. The internal language that the services should use to interact with the bridges is not mentioned in the paper. However, translation and integration of home automation protocols is a non trivial problem, because of different semantics. Also with an approach using a central server, it is more difficult to obtain a balanced business relationship between market actors since one actor is likely gain dominance over the server. Compared to our approach we propose an number of common protocols instead of common middleware and platform.

Service orientation and service composition in pervasive computing environments have previously been explored in [7].

III. Design

In this section, we describe and motivate the design of the HomePort layered architecture. The design of the architecture has been guided by the set of requirements introduced above. To ensure that the requirements are both relevant and realistic, the requirements originate from scenarios developed in cooperation with a number of industry partners.

The architecture of the HomePort system can be divided into four layers: device, bridging, service, and composition. Each layer handles a part of the functionality required to meet the requirements described above. In figure 1, an overview can be seen.

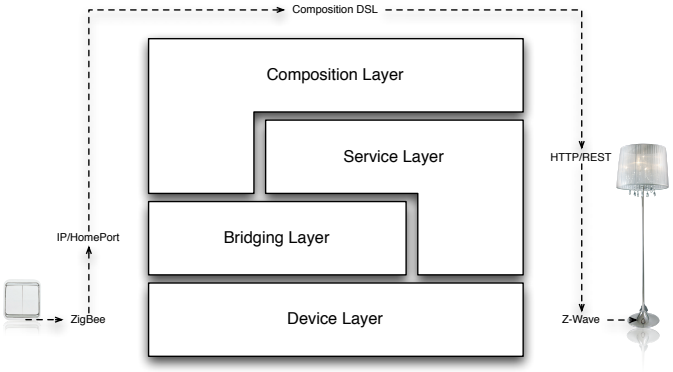

Fig. 1. Architecture Overview

4) Device Layer: The device layer consists of a set of home devices each connected to a subsystem such as e.g. ZigBee or Z-Wave. To ensure that businesses remain in control of their subsystems (and thereby maintain current business models), the HomePort architecture makes no assumptions at the device level. It is, however, clear that to use devices from a particular subsystem, it has to be possible to somehow communicate with the devices.

The devices might belong to any of the traditional home automation domains, e.g. heating and ventilation, entertainment, lighting, security, or entirely new domains.

5) Bridging Layer: In the bridging layer, subsystem devices are made accessible over IP via a Bridge device using the heterogeneous network protocol (h-net). The Bridge relays commands to and from the subsystems using minimal translation or adaption of subsystem protocols. To interact with devices, commands are sent to the Bridge which in turn replies. The bridge can also initiate communication by broadcasting UDP datagrams if relevant events occur in a subsystem.

The bridge layer functionality is split in two parts, a generic IP bridging part (h-net) and a subsystem dependent part. The latter part must have access to radio hardware that is compatible with that particular subsystem. In order to provide a well defined interface for connecting arbitrary subsystem protocols, a *common network adaptor interface* (CNAI) separates the two parts. Market actors can develop homeport compatible network adaptors that implement the subsystem dependent part as an external module to a bridge component.

Since the functionality of the Bridge device is very basic, the requirements on the hardware platform is very limited and therefore the Bridge device can potentially be very cheap to manufacture and consumes an insignificant amount of power.

6) Service Layer: The responsibility of the service layer is to present device functionality to the composition layer in a subsystem independent manner using a common language. A Gateway device is connected to a number of Bridge devices and exposes the devices of the subsystems encapsulated by the bridges through an HTTP based service protocol in the Representational State Transfer (REST) [8] architectural style.

The Gateway is also responsible for enforcing access policies for devices. If, for example, a lamp is repeatedly turned on and off due to a controller loop, it is the responsibility of the gateway to disable access from the controllers to make sure that the bulb is not destroyed. It is left as future work to implement a language for specifying access policies (currently, they are implemented by hand).

Each device is modelled as a web resource that can be interacted with by using the HTTP [9] methods GET and PUT. For example, to read the state of a lamp, a GET request is sent to the URL representing the lamp (e.g. `/services/lamp-42`). In return the Gateway replies with a representation of the state of the lamp. The encoding of the state is specified by the client by including an `Accept` header in the request (e.g. `applicaiton/xml`, `applicaiton/json`, `text/html` or `text/plain`).

Having a REST based service oriented architecture ensures that device functionalities can be composed across subsystem and ownership domains. When interacting with a device at the service layer, it is transparent what type of wireless network it is connected to.

The HTTP protocol is light-weight and implementations can be found for almost any platform. By using the `Accept` header, the client can decide whether the compact and easily parsable `text/plain` representation should be used, or if the more verbose but type safe `application/xml` should be used. This enables small scale systems to interact with home devices while at the same time let the application developer access services through standard APIs. Another benefit of using HTTP during development is that a standard Internet browser can be used to inspect services and interact with them. For the HomePort system, we have developed a simple user interface based on Asynchrounous Javascript and XML (Ajax [10]).

The Gateway running the service platform is implemented in the Python [11] programming language and uses less than 5 MB of RAM.

7) Composition Layer: In the composition layer, applications are formed by combining multiple home devices. This can be done by hand-coding application logic and service invocations or by using a domain specific service composition language [7]. In section IV, we give an example of such a language. To make it possible for users and developers to inspect which services are connected, it is required that the composition mechanism makes this information available to the composition layer.

To support current business models (see section I), an alternate approach can also be used. For businesses producing specialized hardware controllers (e.g. light controllers), there is no value proposition in opening up their domain completely because this would enable third party controller manufacturers to compete Instead, we propose that the HomePort service infrastructure acts as a subsystem to the hardware controller. Hereby the controller manufacturer is in control of what is exposed to the world. This could, potentially, imply that nothing is exposed, but we conjecture that domain specific businesses see an added value in adding functionality from other domains. For example, for a business manufacturing light controllers it would be attractable to integrate with a fire alarm system to enable automatic lighting in emergency situations or a system for monitoring energy consumption.

In some scenarios it may be necessary to leave out some of the layers due to hardware and cost requirements. If, e.g., the user is interested in a setup that allows for custom composition, the Bridge layer can be left out. Hereby, the Gateway node communicates directly with subsystem nodes. If, however, the user is not willing to pay the price of the Gateway, composites can be created without involving the service layer. These composites will, however, have to contain both a bridge level service discovery and a custom translation module that is able to translate the relevant set of commands and events. Currently, bridge level service discovery is left as future work.

In figure 1, a simple configuration is illustrated. A ZigBee switch is connected to a Bridge node that, via the HomePort IP protocol, communicates with a running composite. The composite interacts via HTTP/REST with the service framework on a Gateway node which, in turn, is directly connected to a Z-Wave lamp.

A. Impact on legacy equipment

For the homeport system to have impact on the situation of home automation regarding energy conservation, vendors, in the near future, have to produce homeport compatible devices i.e. controllers, network adaptors, bridges, gateways. In order to make it feasible for vendors to produce and market these devices, we have kept in mind the impact on the design that our system requires. Figure 2 illustrates how different types of devices are affected when adopting to homeport.

The figures shows that end-device are not affected at all and that Homeport compatible products i.e. controllers, bridges and gateway must be fundamentally changed in order to take fully advantage of the homeport system by either implementing

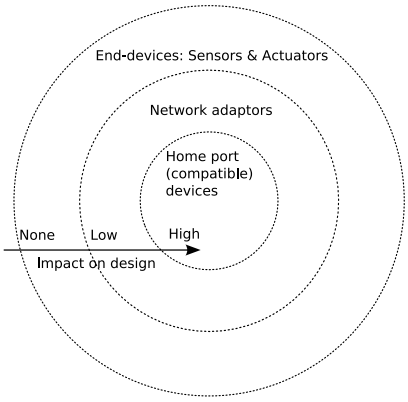

Fig. 2. Impact on design on different device types: the area corresponds to the amount of devices. The larger sensor and actuator group is unaffected by the homeport integration.

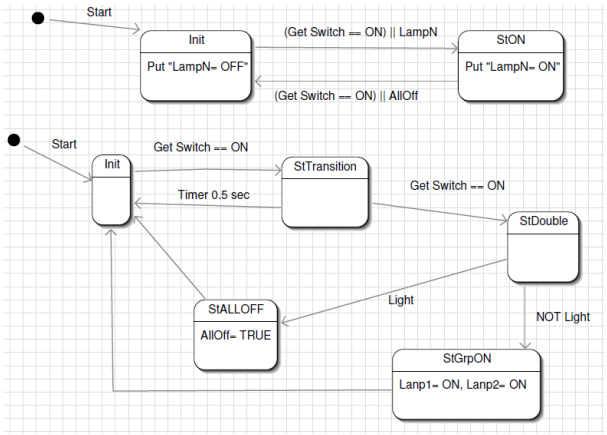

Fig. 3. Two FSM's. Öne for LampN and one for the double push function

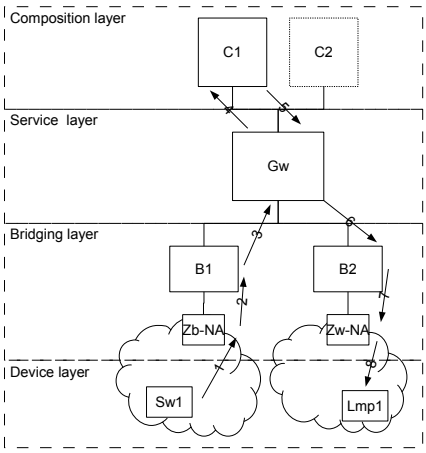

Fig. 4. Prototype system diagram: C1 and C2 are controllers, Gw is a homeport service gateway, B1 and B2 are bridges, Zw-NA is a Z-wave network adaptor, Zb-NA is a Zigbee network adaptor, Sw1 is a zigbee switch, Lmp1 is a Z-wave lamp

the bridge protocol or the service protocol. Between those two groups exist another group of devices that is medium affected, since they today exist as network coordinators that knows how to interact with the wireless network. They must be updated somewhat to interface the homeport system. Changes involve removing the "application layer" part and replacing it with adaptor software that accepts commands from the serial port protocol using the *common network adaptor interface* protocol.

IV. The Homeport Control Logic Language (HCLL)

The control logic is placed in the compositions layer. This layer consists of an interface to the service layer and an interpreter that execute the control logic. The specification of the control logic is based on the language HCLL (Homeport Control Logic Language). The main control structure in HCLL is the definition of Finite-state machines (FSM). A FSM, is a model of behavior which is composed of a finite number of states, transitions between those states, and actions. The transitions are governed by events.

In this section a small example is given: A lamp (LampN) is switched ON and OFF by a push button. If the button is pushed twice (double push) then all lamps in the house should switch off. If all lamps are off then the double push should, as a default light setting, switch Lamp1 and lamp2 ON, i.e. the user might have a number of lamps that are always turn on when he/she comes home.

The service layer interface loop use the blocking call in the service layer to read inputs from devices via the HTTP protocol. These inputs can be events e.g. a button that is pushed, a switch that switch from one state to another or they can be input values e.g. a temperature, a humidity and so on. If the input is an event the corresponding event in the EI data is updated. Likewise, if the input is an input value, the input in the EI data is updated.

The Interpreter loop runs the HCLL program. This loop starts by locking the EI data when executing the HCLL program. This locking secures deterministic behavior of the FSMs. After executing the program all the set events is reset and the EI data is released.

V. Examples

We have made an experiment testing that Z-wave lamps via our system could be turned on by a zigbee switch.

In figure 4, a deployment diagram describes the experiment. The equipment used to implement bridge nodes includes: Zigbee development kit [12] connected to a NSLU2 (B1) using a USB-serial converter and a Z-wave adapter connected to another NSLU2 (B2) via USB.

End-devices (sensor/actuator nodes) equipment include: Z-wave appliance and lamp plug-in modules and another Zigbee development kit simulates a Zigbee switch. The gateway role is exemplified using yet another NSLU2 (GW) with Linux with homeport protocol translation software.

All the bridges and the gateway are connected via LAN network (i.e. UTP network cables and Ethernet switch) and during the experiment the composition logic (C1) was running on the gateway node (GW) using the HTTP service protocol.

Also, for the sake of testing our ideas at the system level, some of the functionality of the *homeport compatible network adaptors* (i.e. z-wave adaptor) was placed on the bridge node (B2) as a driver module. We envision that in the future this module is moved to the network adaptors.

The experiment has shown that if the Zigbee switch was pressed (1) it would send a Toggle-message to the Zigbee network adaptor, that would forward the payload (2) to the bridge using the common network adaptor interface. When the bridge receives a broadcast message from a CNAI it will transmit the payload (3) in UDP broadcast on a predefined port on the LAN. When the gateway node receives a UDP-broadcast it translates and uses the IP and port information to determine which end-device it originates from e.g. 192.168.1.152:9002 = Zigbee switch SW1. Via HTTP streaming commands the gateway is notifying (4) relevant compositions about the toggle-message.

Also the experiment has shown that if a composition via the HTTP interface of the gateway initiates a end-device command, a corresponding message is send to a predefined port on a predefined IP. When a bridge receives a message on port N, the message is forwarded via the z-wave driver to the z-wave adaptor, that in turn sends a z-wave message to the lamp.

Using a (switch-lamp) composition, it was verified that the system was able to turn on a Z-wave lamp when a Zigbee switch was pressed. Despite the added protocol translation overhead, the observed delay, from you press the button on the switch until the lamp turns on, appeared to be instant.

Working with different network technologies and their adaptors, we have discovered that the alternative to enforcing the common network adaptor interface, would not only require one driver per network technology, but a different driver for each network adaptor or stack. The reason for this is that not all wireless network technologies have a well defined standard remote interface, i.e. a serial port interface for sending commands to and receiving event from the wireless network.

VI. Conclusions and Future Work

In this article we have shown how home automation can be based on infrastructure and how multiple protocols such as Zigbee and Z-wave can coexist. The contribution of the paper is an architecture design that is capable of handling multiple protocols and allowing vendors to share interfaces. From our experiments we can conclude that it is possible to translate between Zigbee and Z-wave lamps/switch on on/off basis. Also during this work we have discovered that commercial vendors are willing to partly open their systems, as long as they decide the interface. Each technology and each stack has different serial port interface/protocol, and we have verified the need for common network adaptor serial port interface protocol. Regarding protocol evolution, we have confirmed that a translator gateway only is able to handle the (wireless network) commands that were known at build time, i.e. we have build a translator gateway that can translate on/off commands in different arbitrary protocols, but it is at the time of this writing unable to translate dimmer commands.

Update costs and incompatible hardware platforms used by infrastructure device vendors makes software updating difficult and undesirable in large scale adoption of frameworks in the industry. The framework we have proposed allows an installed HomePort system to be updated by installing new hardware nodes with new functionality such as new types of sensors, gateways and controllers. However, to do this via auto configuration is future work.

Until recently, our focus have mainly been on the lighting domain. To fully realize the environmental and economic potential of being able to control energy consumption in the home, it is necessary to include other domains into the system. For example, context information from an alarm system could be used to turn off heating when nobody is at home. While it is relatively easy to ensure safe operation of devices from the lighting domain, the same cannot be said for all devices in other domains. To be able to ensure safe and correct interaction with other types of end-devices, we plan on designing a language for defining access policies. Such a language will be interpreted by the gateway that will enforce the defined rules.

References

[1] G. Kastner W. Neugschwandtner and H. Soucek, S. Newmann, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.

[2] Zwave Alliance, "Z-wave protocol," http://www.z-wave.com.

[3] Zigbee Alliance, "Zigbee wireless technology (ieee 802.15.4)," http://www.zigbee.org.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practise*, 2nd ed. Addison-Wesley, 2003.

[5] V. Miori, L. Tarrini, M. Manca, and G. Tolomei, "An open standard solution for domotic interoperability," *Consumer Electronics, IEEE Transactions on*, vol. 52, no. 1, pp. 97–103, 2006.

[6] J. Bourcier, C. Escoffier, and P. Lalanda, "Implementing home-control applications on service platform," *Consumer Communications and Networking Conference, 2007. CCNC 2007. 2007 4th IEEE*, pp. 925–929, 2007.

[7] J. Brønsted, K. M. Hansen, and M. Ingstrup, "Issues in Service Composition for Pervasive Computing," Accepted for publication in IEEE Pervasive Computing.

[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC 2616: Hypertext transfer protocol–HTTP/1.1," http://www.ietf.org/rfc/rfc2616.txt, June 1999.

[10] M. Mahemoff, *Ajax design patterns*. O'Reilly Media, Inc., 2006.

[11] M. Lutz, *Programming python*. O'Reilly Media, Inc., 2006.

[12] Microchip.com, "Picdem z," http://www.microchip.com/zigbee.