A Hierarchical Wireless Network Architecture for Building Automation and Control Systems

Mohammad Mostafizur Rahman Mozumdar*, Alberto Puggelli*, Alessandro Pinto†
Luciano Lavagno‡ and Alberto L. Sangiovanni-Vincentelli*
*EECS, University of California Berkeley, CA
{mozumdar,puggelli,alberto}@eecs.berkeley.edu
†United Technologies Research Center Inc. Berkeley, CA
alessandro.pinto@utrc.utc.com
‡Politecnico di Torino, Italy
lavagno@polito.it

***Abstract*—The building automation industry is experiencing a sudden increase in the complexity of control systems, mainly due to the push towards energy efficient buildings. These systems are necessarily distributed and rely on a communication network to gather data from sensors, produce intermediate results, and send commands to actuators. These networks should be cost effective, and should be flexible enough to be easily reconfigured if the building usage changes over the years. Wireless sensor and actuator networks are, therefore, key enablers. In this paper, we propose a hierarchical wireless network architecture for building automation and control systems and a protocol to manage it. We implement gradient based routing (for collecting data) and label switching table (for disseminating configuration commands), thereby supporting upstream and downstream data flows across the network.**

***Keywords*-Sensor Networks, Building Automation Systems, Implementation**

I. Introduction

Residential and commercial buildings in the United States are responsible for up to 73% of the total energy consumption [14]. To reduce the energy consumed by the building stock, both new constructions and existing buildings must be equipped with energy efficient solutions. These solutions are not only architectural, but rely on the use of advanced control algorithms that, based on measurements collected by sensors, compute an optimal control policy and send commands to actuators. Thus, the sensor-actuator network is a key element of building automation systems for energy efficiency. The selection of an optimal network is driven by several concerns including cost. Installation cost is one of the major concerns in building retrofits mainly due to wiring. For new constructions, although wiring is still a concern, the major problem is in making sure that the network is flexible enough to accommodate changes in the building usage that are common over its life-cycle.

For these reasons, wireless networks have been always considered as an interesting technology. Wireless sensor networks (WSN) find applications in factory and building automation, environmental monitoring, security systems and in a wide variety of commercial and military systems. However, wireless networks operation cost (which is mainly due to battery replacement) and low reliability have long been the major roadblocks to their adoption. Moreover, the whole-building control policy is often hierarchically structured: room-level controllers communicate to zone-level controllers, which in turn communicate to floor-level controller and so forth up to the building management system. This structure follows the geometry of the building. At each level, some computational power is required to execute control algorithms which is a challenge in the case of wireless sensor networks.

Based on these observations, we argue that the right networked system for building automation is in between a wired and a wireless solution. The first contribution of this work is the selection a design point that seems to be a good compromise between these two extremes. We distinguish between the power network and the data network. The proposed solution is entirely wireless for what concerns the transmission of data. However, we propose a hybrid approach for the power network where sensors are battery powered and are used only for measurement of physical quantities; actuators can be battery powered or not (a power source might be available close to them); and the rest of the components are powered by a wired network. These components provide also some computational power to execute control algorithms. This architecture responds to the needs of being able to reconfigure the sensing platform without major efforts, and to support the structure of a building control system, namely hierarchical and based on building geometry. The second contribution of this paper is a protocol to manage the network so that devices can be easily plugged into the systems, and the network is robust and reliable. The proposed protocol is based on a minimal set of APIs assumed to be provided by the lower layer of the stack. Thus, the proposed protocol can be implemented on other standard protocols such as Zigbee [11]. The third contribution of this paper is to illustrate the implementation details of

 ISBN: 978-1-61208-133-5

a hierarchical sensor network taking Building Automation System (BAS) as a context, so that the application engineers for BAS can use the proposed protocol implementation directly or can customize it according to their specifications.

The remainder of the paper is organized as follows: in Section 2, we review related works including existing standards, and we present some existing routing protocols that could be suitable candidates for building automation. In Section 3, we introduce our proposed architecture and in Section 4, we describe the network formation and maintenance protocol in details. We present the modeling and evaluation of the architecture in Section 5. In Section 6, we provide a set of conclusion and our future research directions.

II. Related Work

Building automation systems went through different phases of sophistication, from point-to-point centralized implementations to more complex networked systems. The implementation of these systems rely today mainly on wired networks. EIB or its successor KNX [10], LonWorks [13] and BACNET [12] are among the most used standard protocols.

Recent advancements in wireless and sensor technologies have pushed forward new emerging protocols and standards such as ZigBee, Z-Wave[15], WirelessHART [16], and others. In general, the deployments of WSN based systems is *convergecast* [9][2], in which a cloud of sensor nodes transmit data to one sink. Sensors are connected to the sink over a mesh or tree network. Since mesh networking is in general resource hungry, tree networks are typically used. To send configuration commands to sensor nodes, one of the most widely used mechanism is controlled *flooding* [4], [1] which is the simplest solution to implement but it has several drawbacks in sparse networks. A more complex solution is source routing [5], in which the sink defines the route of the command packet based on its routing table.

Little attention has been paid to defining wireless protocols and architectures that are suitable for building automation systems, which is one the main motivations of this work. We put particular emphasis on the network organization which could become the underlying layer of any wireless standard. Recently, the IETF work group ROLL [1] has selected gradient based routing for collecting data from sensor networks [6] and we envision that a customized variant of this technique could be a good fit for BAS.

In gradient based routing [3], the backbone network is initially formed by using controlled flooding. The first node that starts the network formation broadcasts a message by setting the gradient height to 1. Nodes that receive gradient messages set their gradient to the received value, and broadcast a gradient value which is equal to their gradient plus 1.

[1]The goal of the IETF ROLL is to standardize a routing protocol for wireless sensor networks.

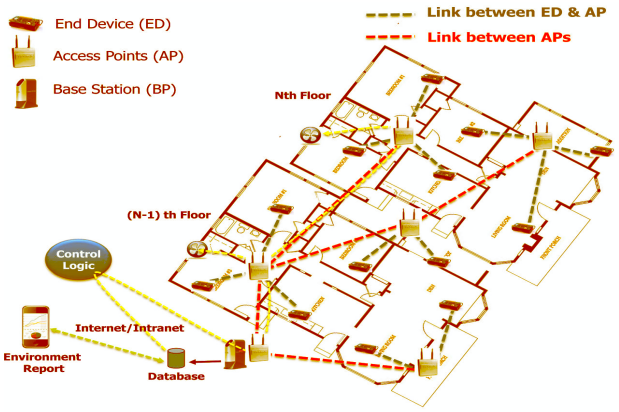

Figure 1. A snapshot of the BAS

This constructs a network organized into layers where the higher gradient nodes report to the lower gradient ones. After network formation, one of the main concerns is to maintain the gradient levels at each node. In fact, wireless links may appear or disappear at any time which may cause problems [8]. If a link disappears, then a node might have to select a new parent among a set of reachable ones. In this cases, a node always joins the network so to reduce its gradient level [7].

III. An Overview of the Architecture

An example instance of the proposed architecture is shown in figure 1. The network architecture is organized into a hierarchy of components that include end devices, access points, base station, and a server connected to a database. The role of each component in the network is the following:

- An **end device** is a sensor or an actuator. Both components can be battery powered. Having battery powered sensors give the flexibility of quickly relocating them. Each device has a *floor* and a *room* ID and can join in the network through any access point on the same floor (in a star configuration), which also provides for in-network load balancing. A sensor node could be configured for periodic or threshold crossing reporting depending on the quantity to be measured (e.g temperature, light and occupancy). To conserve battery power, sensor nodes should sleep most of the time and should wake up periodically to acquire data from sensors and transmit them to the associated access point. On the other hand, actuator nodes (which drive actuators), should periodically interrogate the access point for any pending command from the base station and then tune the actuator based on the received configuration.
- An **access point** is part of the backbone network that is used for data collection and command routing and is connected to direct power supply. These devices can be always on and capable of low power listening

to minimize energy consumption. Each device has a *floor* ID and a network-wide unique ID (for routing, as described later in this paper). These devices permit end devices to join the network and send data to collection points, construct aggregated packets and route them to the base station. The base station uses these nodes to configure sensors and actuators.

- A **Base station** has wireless connections with access points and an Ethernet connection for LAN or web access. A base station works as a master and initiates the formation of the backbone network. It collects the sensor data and logs them into the database which can be analyzed for trending and optimization. There could be one or more base stations for the whole system depending on the network size.

The architecture also includes server, storage database, control logic and web applications to control and analyze the behavior and data of the deployed networks, because of the space limitation we are excluding the details of these components from this paper.

IV. Network Formation and Maintenance

The network is formed and activated by following a series of phases. First, the backbone network is formed. The base station and the access points participate to this phase. Then, end devices join with access points. Each access point sends a report of the connected end devices to the base station. The control logic sends configuration commands to activate/deactivate sensors and actuators. After activation, end device sensors periodically report to the associated access point node which aggregates sensor data to construct a single packet that is routed to the base station. The following sections describe these phases in detail, including the issue of node failure.

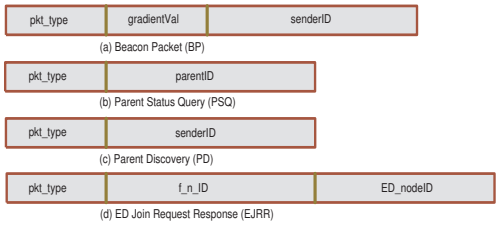

Figure 2. Packet format for network formation and maintenance

A. Backbone network between base station and access points

The main purpose of the backbone network is to support routing of messages in the network. We choose gradient-based routing to form the backbone. The formation of the backbone network is initiated by the base station which constructs and broadcasts the Beacon Packet (BP) shown in figure 2(a), with *gradientVal=1* and *senderID = BaseStationID*. Initially, all active access points set their parent to *NIL*, and the gradient to a very high value. Access points that are in the radio range of base station receive the BP and set their gradient to 1, and their parent to the *BaseStationID*. After receiving the BP, an access point updates its gradient if any of the following holds:

- It has no parent.
- It has a parent but BP is received from an access point node that has lower gradient.
- It receives the BP from its existing parent but the parent's gradient has been changed (this scenario might arise when an access point is added or deleted in the network, as described in detail later).

An access point updates the values of its *gradient* and *parent* taken data from *gradientVal* and *senderID* of the BP respectively. An access point always broadcasts the BP after updating its gradient and/or parent value. When broadcasting, an access node modifies the BP by incrementing the gradient by one, and by setting *senderID* to its own ID. While broadcasting the BP, the node waits for a random time and uses a simple CSMA/CA protocol at the MAC layer to reduce collisions. This process of controlled flooding continues until the backbone network is formed.

B. Tributary network between access points and end devices

After power-up, each end device constructs and broadcasts the EJRR (ED Join Request Response) packet shown in figure 2(d). While constructing the EJRR packet, end devices set *f_n_ID = floorID*, *ED_nodeID = nodeID*, and the value of *pkt_type*. The *pkt_type* field can take three different values (used for the end device joining process) that are listed below:

- pkt_type = requestVal, when the end device broadcasts a join request
- pkt_type = responseVal, when the access point sends a response to the end device
- pkt_type = confirmVal, when the end device sends confirmation of joining to the access point

Any access point node after receiving the EJRR packet checks the *f_n_ID*. If the end device joining request is coming from other floors it ignores the request, otherwise it modifies the EJRR packet by changing the value of packet type (responseVal) and by setting *f_n_ID = its nodeID* and then rebroadcasts it. An end device might get multiple responses from different access points, but proceeds with the first response and sends a confirmation EJRR packet by simply changing the value of packet type (confirmVal). The specified access point (by checking the *f_n_ID*) adds the end device into its children list and routes all data from it to the base station. It also sends the configuration packets from the base station to the end device.

C. Handling access points node failure

Since access points create the backbone network, and there is only one path from any access point / sensor

node to the base station, if a link becomes unavailable between two access points the network can be disconnected. That's why each access point checks its parent status at a periodic interval. After choosing a parent, each access point sends a PSQ (Parent Status Query) packet shown in figure 2(b) at random *nQT* interval. In every *nQT* interval, an access point constructs a PSQ packet with *pkt_type* = *psq_request*, *parentID* = *its parentID* and broadcasts it. After receiving the PSQ packet, the parent of the requesting access point sends a response by simply updating *pkt_type* = *psq_ response*. This response might be received by all or some children (access points) including the requesting one. The requesting node schedules an event at current time plus its *nQT* to check its parent status again, and all other child nodes, who have heard the response postpone their immediate PSQ and schedule another future event to check the status of their parent. If a child does not receive the PSQ response from its parent for three consecutive times, it sets its gradient/parentvalue to NIL and searches for a new parent.

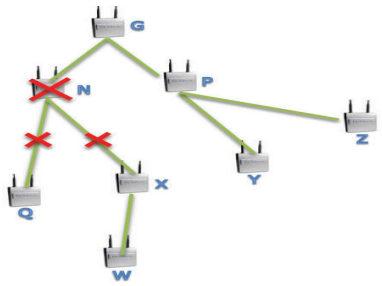

Figure 3. Example network setup for searching new parent

D. Searching for a new parent

Searching for a new parent is required when a new access point is added to the network and when the children of a dead access point search for a new parent, as described by the example shown in figure 3. Let us consider the case where access point node N dies and node X has failed to get a PSQ response for three consecutive times (same for node Q, but let us assume that the *nQT* timer of node X expires first). Node X constructs the PD (Parent Discovery) packet shown in figure 2 (c) with *senderID*=*X*. Suppose that this PD packet is received by P, Q, Y and W and they proceed as follows:

- A node after receiving a PD packet checks its *senderID* field first.
- If *senderID* = *Node's parentID* then the node does nothing (e.g. node W). This prevents loop formation.
- If *senderID != Node's parentID* then the node checks whether its parent is alive or not (by sending PSQ packet). If its parent is not alive, then the node does not do any further processing (e.g. node Q). This step prevents following a dead route.
- If the node's parent is alive, the node simply broadcasts a BP (e.g. nodes P and Y).

Node X might receive two BPs, either from P first, and then from Y, or vice versa.

CASE 1: First P then Y

- **BP from Node P:** Node X sets P as its parent and changes its gradient value. Since the beacon's gradient value is modified, node X broadcasts it. The beacon packet from X might be received by nodes Q and W. Since node Q is also looking for a parent, it sets X as its parent and broadcasts the beacon packet. Node W, after receiving the beacon packet form X, finds that the beacon is from its parent. If the gradient has been changed, node W modifies its gradient and broadcasts it again.
- **BP from Node Y:** Node X finds that the beacon packet contains a higher gradient value, so it does not modify its gradient value.

CASE 2: First Y then P

- **BP from Node Y :** Node X sets Y as its parent and changes its gradient value. Since the beacon's gradient value is modified, node X broadcasts it. The BP from X might be received by nodes Q and W. Nodes Q and W follows the same procedure as described before.
- **BP from Node P:** Node X finds that the beacon packet contains a lower gradient value, so it changes its parent to P and broadcasts the beacon packet. Nodes Q and W update their gradient value but their parent remains the same (Node X).

Both cases lead to the same network setup. Let us consider the scenario of node Q that is also looking for a new parent approximately at the same time as X.

- Node Q broadcasts the PD packet which might be received by node X and W.
- Suppose that node X is still in the process of finding its parent (currently it's parent is not alive, so it will not respond).
- Since Q is not W's parent and the parent of W (node X) is alive, W broadcasts the beacon packet which might be received by both parentless nodes Q and X.
- Both Q and X set their parent as W, which leads to formation of an isolated graph (it could also be a loop if X sets Q as a parent)
- When node X receives the BP either from node P or Y, the loop is broken and graph becomes connected again. A node needs to send the PD packet after knowing that its parent is not alive, even though it receives a BP earlier than sending the PD packet.

E. Upstream and downstream

Each access point collects sensor data from sensors and *upstream* flows route these data to the base station. Since

 ISBN: 978-1-61208-133-5

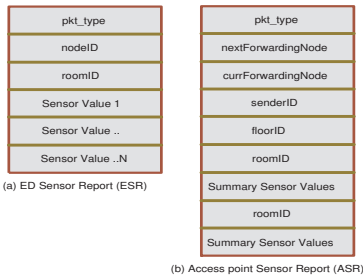

Figure 4. Packet formats for sensor data reporting and routing

each access point has a unique parent in the backbone tree and the base station is the root of that tree, data from each access point follows a unique path to the base station. Whenever a new end device (sensor/actuator) joins an access point, it immediately sends a notification report to the base station. If needed, the base station sends configuration commands to the end devices using the *downstream* flow. After configuration, each end device constructs ESR packet (shown in figure 4(a)) periodically or after crossing a specified threshold, and sends it to its access point. The ESR packet contains room identification and also collected sensor values. The access point node constructs an ASR packet (shown in figure 4(b)) at periodic intervals and sends it to the base station. While constructing the ASR packet, the access point node puts the following routing information in the header:

- nextForwardingNode = Its parent ID,
- currForwardingNode = Its NodeID
- sender = Its NodeID

Whenever an ASR packet flows along the upstream path to the base station from the originating access point, the intermediate access points change the value of *nextForwardingNode* and *currForwardingNode* accordingly and record the following information:

- Sender NodeID (sender field of the ASR packet)
- Immediate downstream router ID (currForwardingNode field of the ASR packet)

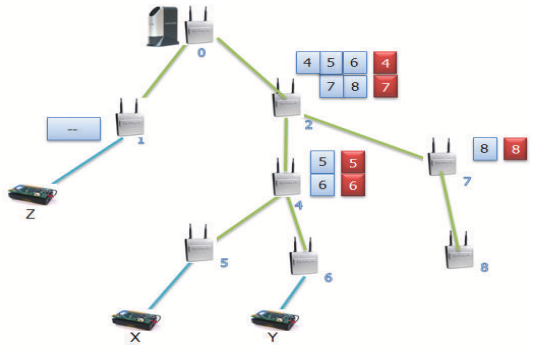

Figure 5. Upstream and downstream routing

These information are needed for downstream command/configuration packet routing. Let us consider the simple network setup shown in figure 5. In the case of upstream flow, node 5 sends its data to the base station through nodes 4 and 2. Nodes 2 and 4 update their downstream routing table for node 5. Whenever node 2 receives a command/configuration packet for node 5, it simply routes it to node 4, and node 4 routes it to node 5. The same process applies to all other intermediate routing nodes. In the up/down stream routing, each access point uses its parent node for upstream routing and the one-hop label switching routing table for downstream routing.

V. Modeling and Evaluation

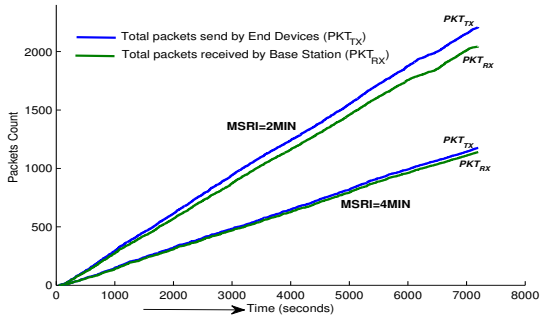

Figure 6. Packet loss count in the whole network

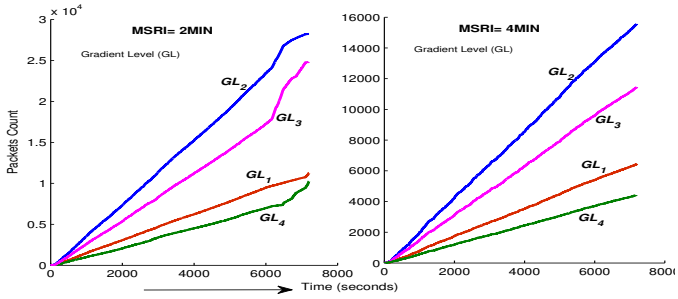

Figure 7. Traffic load at different gradient level

We developed an OPNET model for the proposed architecture. OPNET has three hierarchical component levels: the network level creates the topology of the network, the node level defines the behavior of the node and controls the flow of data between different functional elements inside the node, and the process level describes the underlying protocols by using finite state machines (FSMs). We developed three kinds of nodes, as described in our architecture, with the routing logic described in Sections III and IV. For the evaluation, we created the first setup with 1 Base station, 18 Access Points and 21 End device nodes deployed in a seven-storey building. We configured two scenarios for collecting

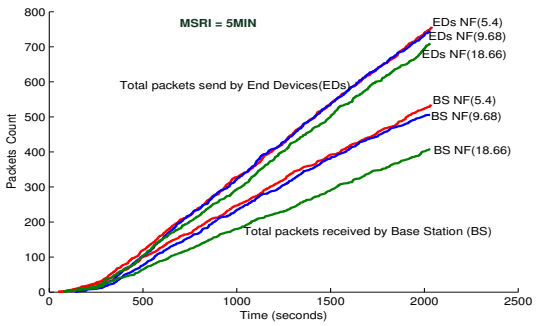

Figure 8. Packet transmissions and receptions at different Node Failure (NF) rate

data from end devices by setting the MSRI (Max Sensor Reporting Interval) at 2min and 4min respectively (The base station sends these configuration commands when the backbone network has been formed and end devices joined into the network). Thus, sensor reporting will be scheduled at every $uniform_{dist}(1.00) * MSRI$ interval. Figures 6 and 7 depict the packet loss and traffic load characteristics of the example network setup. The results show that at the higher interval (MSRI=4min) the network has less packet loss. To minimize the height of the backbone tree, we put the base station at the 4th floor (near the middle of the building), so the the backbone tree expands both up and down. This is reflected in figure 7 which depicts traffic loads (both sending and receiving packets) at different gradient levels. Since the base station (gradient value 1) most of the time receives packets but sends only a few configuration packets, its load is less than that of the 2nd and 3rd level nodes. To understand the effect of random node failures on the proposed architecture, we created another network setup with 1 Base station, 75 Access Points and 100 End device nodes deployed in a fifteen storey building. Figure 8 depicts the loss of packets at different node failure rates (NF (nodes/sec) = 5.4, 9.68 and 18.66). Here a node fails at any random time, then revives and joins back in the network. Figure 8 depicts that the packet delivery ratio remains relatively the same for NF rates up to 13% (NF=9.68) and afterwards it drops gradually. This means that the proposed architecture can tolerate 13% of access points failure in the most extreme case. Since the access points are directly connected to the power line, such high node failure rate is unlikely (we expect software/hardware malfunction to be rare).

VI. Summary and Conclusions

We presented a hierarchical wireless network architecture for building automation and control systems, and a protocol that efficiently solves the problems of forming and maintaining the network. The proposed architecture is hybrid with respect to power: some nodes are battery powered and others are plugged into a power supply. This hybrid approach provides a good trade-off between reliability and flexibility. We modeled the network components and protocol using OPNET. Our future work includes tools to generate application code for each node in the network, which entails generating drivers for the underlying low level protocol to implement the protocol primitives presented in Section IV. We will target standard platforms such as TinyOS and ZigBee.

References

[1] D. Braginsky and D. Estrin, Rumor routing algorthim for sensor networks, In *WSNA '02*, pages 22–31.

[2] T. Chen, H. Tsai, and C. Chu, Adjustable convergecast tree protocol for wireless sensor networks, *Computer Communications*, 33(5):559–570, 2010.

[3] J. Faruque, K. Psounis, and A. Helmy, Analysis of gradient-based routing protocols in sensor networks, In *DCOSS*, pages 258–275, 2005.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, In *MobiCom '00*, pages 56–67.

[5] D. B. Johnson and D. A. Maltz, Dynamic source routing in ad hoc wireless networks, In *Mobile Computing*, pages 153–181, Kluwer Academic, 1996.

[6] R. D. Team, Rpl: Routing protocol for low power and lossy networks, In *IETF ROLL WG, IETF Internet-Draft*, 2009.

[7] T. Watteyne, I. Augé-Blum, M. Dohler, S. Ubéda, and D. Barthel, Centroid virtual coordinates - a novel near-shortest path routing paradigm, *Journal of Computer and Telecommunications Networking*, 53(10):1697–1711, 2009.

[8] T. Watteyne, K. Pister, D. Barthel, M. Dohler, and I. Auge-Blum, Implementation of gradient routing in wireless sensor networks, In *GLOBECOM*, pages 1–6, 2009.

[9] H. Zhang, A. Arora, Y. Choi, and M. G. Gouda, Reliable bursty convergecast in wireless sensor networks, In *MobiHoc '05*, pages 266–276.

[10] KNX Association, 2003, `http://www.knx.org/`

[11] Zigbee alliance, 2004 `www.zigbee.org/`

[12] BACnet, A data communication protocol for building automation and control networks, 2003, `www.bacnet.org/`

[13] Echelon, Lonworks technology overview, 2002, `http://www.echelon.com/`

[14] EIA, Annual energy outlook 2009, technical report, doe/eia-0383 (2009), In *US Department of Energy, 2009*.

[15] , Z-wave alliance, 2007, `www.z-wavealliance.org/`

[16] WirelessHART, Hart communication foundation, 2007, `http://www.hartcomm.org/`