Design of Home Energy Gateway Boosting the Development of Smart Grid Applications at Home

Raffaele Bolla *, Maurizio Giribaldi †, Rafiullah Khan ‡ and Matteo Repetto †

* DITEN Dept. University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy, Email: {raffaele.bolla, rafiullah.khan}@unige.it
† Infocom s.r.l, Piazza Alessi 2/7, 16128 Genoa, Italy, Email: maurizio.giribaldi@infocomgenova.it
‡ CNIT, Research Unit of University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy, Email: matteo.repetto@cnit.it

Abstract—Smart Grids provide an effective and efficient management of production, transmission, distribution and usage of electricity. One of the main issues in designing applications for Smart Grids is the need to cope with a plethora of different devices, communication protocols and data representations. This paper presents the design of a Home Energy Gateway (HEG) that abstracts heterogeneous devices and communication protocols and can be used to develop intelligent applications following a technology-agnostic approach. It is especially conceived to take over efficient energy management at home. Main design issues of our framework are the ability to automatically discover available devices and appliances, to track their operational status, to support different communication protocols and paradigms and to run concurrently multiple services. As an example of service, we address the implementation of a Network Connectivity Proxy (NCP), which save energy by letting unused ICT devices to remain into low-power states without losing their presence on the network. Further, some preliminary results based on the aforementioned service are also presented.

I. INTRODUCTION

The concept of Smart Grid was recently proposed to bring more intelligence into the power grids by massive integration of Information and Communication Technologies (ICT); that provides many advanced services that were never imagined before [1]. Smart Grid requires active participation of users for efficient load management, integration of renewable energy sources, smart metering, smart home or building automation etc [2], [3]. Such services usually require an intelligent device e.g., Home Energy Gateway (HEG) that communicates with the grid and all local appliances (e.g., computers, TVs, refrigerators etc) inside home through wired/wireless technologies [4].

Many Smart Grid applications/services are expected to evolve on the demand side for efficient energy management. This paper presents the design of an efficient and flexible HEG that can boost up the development of new services/applications at the consumers side. It addresses the main building blocks of the HEG architecture and also the required device/appliance characteristics. The HEG architecture is designed to allow independent development and integration of new services and communication modules. It decouples the service logic from the communication interface in order to support heterogeneous devices and networking interfaces. The HEG can offer fast development of different type of services e.g., Home Automation (HA), Network Connectivity Proxy (NCP), Power Management (PM) etc. This paper specifically targets the implementation of a NCP service as an example. The NCP is the approach in which a proxy maintains the virtual presence for high power devices and allows them to enter into low power sleep mode when idle [5], [6].

The rest of the paper is organized as follows: Section II presents briefly the related work. Section III presents some basic services that can be offered by HEG. Section IV describes in detail the reference architecture for HEG. Section V presents the reference architecture of NCP service. Section VI presents preliminary test results. Finally, Section VII concludes the paper.

II. RELATED WORK

The generic design for HEG based on Open Service Gateway Initiative (OSGI) is presented in [4], [7] and [8]. However, they focus on the generic architecture with discussion about the basic set of components required for designing an application. In [9], a novel low cost HEG solution is proposed for smart metering service. It provides real time display of home energy consumption on the television along with daily, monthly and annual energy consumption statistics. The HEG design for developing a novel home energy management system (HEMS) is proposed in [2] that optimizes the energy consumption of home appliances. The proposed system embeds necessary energy management strategies and offers interoperability between different devices. A low cost and reliable HEG controller design for HEMS is presented in [10] that also embeds a plug & play mechanism. An energy efficient sensor controlled home system using DLNA ZigBee architecture is presented [11]. The proposed architecture allows interconnection between two networks and provides plug & play feature between one another’s appliances. A power saving system based on energy-aware control elements in the ubiquitous home network is presented in [12] that embeds visitors identification feature, Internet service, IPTV service, VoIP service and home automation service. [3] presents an architecture for Smart Grid end-user services that facilitates end users to reduce energy consumption, enables better balance between demand and supply and allows new actors to join new services.

This paper presents a very flexible design of HEG that offers independent development and integration of new services and communication modules and efficiently cope with a plethora of different heterogeneous devices and data representations.
The main design issues of proposed HEG architecture are to run concurrently multiple services, provide seamless and automatic discovery of devices of interest for each specific service, track operational status of registered devices and support different communication protocols and paradigms which are accessible by each service.

III. BASIC HEG SERVICES

The HEG can be designed to support different type of services. Some basic services are briefly described below:

A. Load Management service

To avoid unexpected black-outs or load-shedding, utility companies can use dynamic energy pricing. High energy prices are applied during peak demand periods which will encourage consumers to reduce their load in order to save money on their electricity bill. Thus, the HEG can offer Load Management (LM) service that will receive energy price updates from utility companies. It will enable consumers to prioritize all the electrical loads and schedule the low priority loads to off-peak hours when the energy price is low. Further, the utility company can offer different prices to different zones (e.g., residential or industrial) based on their energy demand and peak hours.

B. Home Automation service

The HEG can offer HA service to reduce the energy waste by automating the operations of all appliances according to user specified configuration. Thus, all appliances are automatically activated and deactivated at their pre-specified times. The appliance operation can also be controlled by deploying different sensors at home. Based on the data received from sensors, the HEG can decide when to activate a particular device.

C. Advanced Metering Infrastructure service

The HEG can offer Advanced Metering Infrastructure (AMI) service that automatically retrieves the energy consumption, diagnostic and status data from meters and communicates it to the utility companies using two way communication channel. Further, the AMI service may retrieve demand response pattern and current energy price from the utility companies.

D. Distributed Power Management service

The HEG can offer Distributed Power Management (DPM) service that efficiently integrates the small and micro electricity generating systems. The DPM service usually exploits renewable sources (sun, water, wind) or cogeneration (combined heat and power). The amount of energy produced by the renewables is unpredictable, thus the DPM service may also include the energy storage which stores electricity in other forms of energy.

E. Network Connectivity Proxy service

The HEG can offer NCP service that reduces the network energy waste by forcing the devices to stay in low power modes during idle periods [13], [14]. It wakes them up only when their resources are required. This paper presents the implementation of NCP service as an example of services offered by the HEG.

IV. THE HEG REFERENCE ARCHITECTURE

The HEG provides a set of services to client devices that may be running concurrently and providing a rich set of features. The basic reference architecture of HEG consists of five main components: Protocols, Dispatcher, Devices, Services, and TimeScheduler as shown in Fig. 1. A set of different protocols can be used for communication with remote devices in both directions and are not statically bound a specific local service. The Dispatcher and the local representation of Devices provide the logical abstraction needed to separate Protocols and Services. The Dispatcher delivers messages received from remote devices by the specific local Protocol to the Service in charge of handling it. It is responsible for triggering an operation when some external event occurs. Devices hold data about real devices and take care of sending message from Services to remote devices. The TimeScheduler component provides a generic framework for scheduling unsolicited tasks (e.g., periodic events) at a given time instant. Fig. 1 also shows the main logical flow of information. Red arrows show the path for incoming messages, i.e., messages received from remote devices. The green arrow show information generated by events scheduled at given time instants. Blue arrows show the path for outgoing messages, i.e., messages sent to remote devices. The bidirectional blue arrow suggests some data about devices are stored locally and can be accessed directly without the need for sending messages. Such local data may be updated by different mechanisms (publish-subscribe, notification, polling).

Zooming out the level of details, the HEG architecture looks like in Fig. 2 that also includes other important components to interface the HEG with the outside. The Configuration Database holds application-wide configuration settings, which are made available to all other components (Dispatcher, Protocols, Services etc). The AppManager controls the whole application, providing the user with the ability to manage the application behavior in real-time. It gives access to current information about registered Devices, subscribed events in the Dispatcher, running modules (Protocols, Discoveries and Services). Different User Interfaces are provided to access the AppManager both locally (i.e., by console) or over the network (both with TCP and UDP protocols). The main functionalities
of HEG architecture are described below.

A. Communication with external devices

The communication part of the HEG architecture is indeed made of two components: Discovery and Protocols. Discovery allows finding devices in an automatic way and registering them into the system. It can use different mechanisms: Universal Plug & Play (UPnP), SDP (Bluetooth), SLP, etc. Some of them will be tailored for communication with different local devices, whilst others will scale to the Wide Area Network (WAN) for communication with external entities e.g., utility companies. The device registration also includes the Protocol and Channel to be used for successive communication with it. The Discovery should also announce the HEG capabilities by means of the semantic of the Protocols it manages. The main purpose of Protocol is the translation between system messages and protocol-specific syntax and semantics. The implementation of protocols allows communication with different types of devices on the same or different media.

B. Internal communication

The Dispatcher handles messages and delivers them to the proper service. Its main purpose is to decouple the communication protocol and the HEG’s services. The Dispatcher deals with message priority and inter-services issues (like multicasting). Fig. 3 shows the logic behind the Dispatcher concept. Basically, the function of the Dispatcher is running a set of given functions where some events occur. This requires prior registration of the functions and associated events (the pink table); e.g., two entities register their functions \( f(.) \) and \( g(.) \) for the same event \( E \). Once the events occur, they are delivered and queued in the Dispatcher (the yellow queue); the Dispatcher processes them one-by-one by running the corresponding functions previously registered \( (f(.) \) and \( g(.) \)\). The same event \( E \) is passed as argument to the functions, as it may embed information. The Dispatcher is used to deliver events received from remote devices to the local Service instances.

C. Abstraction of devices

The Device structure is representative of physical devices that access the HEG services. Each Device is supposed to implement a number of different functionalities; which are independent of the services implemented by HEG, but could be fundamental for the realization of specific functions. Different functionalities in the HEG may be managed through different protocols; further, not all Devices are expected to have the same set of functions. Fig. 4 separates logical functionalities into different DeviceModules. Each DeviceModule consists of a number of state variables which represent current device’s status and are continuously updated by suitable notification methods. Protocols are in charge of managing the continuous update of state variables. To save memory, specific DeviceModules are allocated dynamically depending on the actual device’s capabilities as large number of devices may be present embedding a large number of DeviceModules. Each device module includes its own communication Channel to be used for successive communication with that specific device.

D. Abstraction of services

Service is the abstraction of a generic functionality provided by the HEG. Each Service provides a set of operations targeting a specific feature. Depending on the Service type, it may register for different kind of events at the Dispatcher. These events are delivered/received from abstract devices by mean of the Protocols.

E. Scheduling of activities

The basic reference architecture for TimeScheduler is shown in Fig. 5. It is internally organized as a sorted queue of waiting tasks, each with its own due time. Each task is executed once the schedule time is reached. The TimeScheduler also supports periodic tasks. To provide better flexibility, the TimeScheduler allows both scheduling and withdrawing events, as well as checking if an event is already scheduled and the number of occurrences.
F. Internal messages

Messages are exchanged among different components in the HEG and client devices. They carry information and data about specific services and features. Messages can be classified into three main types: (i) Notifications which are unidirectional and unsolicited messages and do not require a response, (ii) Requests which ask for some processing to the entity they are intended to, (iii) Responses which are solicited messages, as they come as an acknowledgment of the processing required by Requests. Notifications are ‘stand-alone’ messages, while Requests and Responses should always come in couple. Notifications are mainly used to update the value of state variables, to inform about the occurrence of specific events, and so on. Protocols are also required to generate Responses if they do not receive any feedback from the remote device (in such case, the Responses will signal a failure) i.e., the internal communication must implement a reliable mechanism.

V. THE NCP SERVICE

The main objective of HEG is the efficient energy management at homes. This is mostly critical due to large number of installations with tens, hundreds or even thousands of small devices. Many of these devices will not likely be always active; they may be activated only for small periods of time when needed (including both, the appliances and sensors). Thus, the NCP would provide an effective and efficient way to exploit low-power states that are and will be present in these devices [15]. The basic requirements for the NCP service are presented in [5]. The reference architecture of the NCP service is depicted in Fig. 6. When the NCP service is started, it registers with the Dispatcher for four kinds of events: NCPActionRequest, RemoveDeviceModule, RemoveDevice, UpdateVariable. The first event is explicitly defined for the NCP service, while the others are generic within the HEG architecture. NCPActionRequests should be sent by Protocols whenever a device requests a NCP service. Every time the power state of the device changes, the NCPService is notified and it activates/deactivates the NCPActions registered by that device [16]. The NCP basic prerequisites are briefly described below:

A. NCP Actions

NCPActions are tasks the NCP accomplishes on behalf of sleeping hosts. Basically, NCP performs three different types of activities: (i) Network Presence e.g., answering to network protocols (e.g., ARP, PING, DHCP etc) on behalf of sleeping hosts, (ii) Heart-Beating i.e., generating routine protocol and application specific messages on behalf of sleeping hosts to keep connections alive, (iii) Wake-up when a packet requires sleeping host’s resources e.g., new connection attempt. The basic actions our NCP provides so far include: (i) ARPRelayAction that answers the ARP request packets on behalf of sleeping hosts, (ii) PingReplyAction that answers the ICMP PING request packets on behalf of sleeping hosts, (iii) DHCP RenewAction that periodically renews the IP addresses of sleeping hosts with DHCP server, (iv) Wake-On-Connection (WOC) that wakes up the host when a new connection attempt at specific protocol and port is received, (v) Wake-On-Packet (WOP) that wakes up the host when a received packet matches the specified pattern, (vi) Send-Reply-on-Packet (SRoP) that sends a given reply on behalf of sleeping hosts when a received packet matches the specified pattern, (vii) TCP Keep Alive Action that maintains TCP connectivity by answering the TCP connection keep-alive messages on behalf of sleeping hosts.

B. Packet Filters

PacketFilters are classification engines that take currently active NCPActions and look for packets matching certain criteria (e.g., source and destination IP addresses and port numbers, protocol, protocol-specific flags and options or packet content). Briefly, it looks for packets matching a single action, a set of actions or all actions. Once a match is found, single or multiple NCPActions associated with that filter are invoked immediately. The PacketFilters run on multiple interfaces and require methods to start and stop the filter, updating the current rule set, adding and removing interfaces and getting information (filter name, filtering string). Only active rules are loaded into the filter, namely those associated to sleeping hosts.

C. Traffic Diversion

Since the NCP takes the place of sleeping hosts by acting on their behalf, the traffic addressed to such hosts needs to be diverted towards it. To aim this, a special PacketFilter is automatically set up by the NCP service at the time the first NCPAction is registered by a device. The ARPRelayAction answers to local ARP requests for the IP address of the device and provides the local MAC address of the device hosting NCP service. Thus, all subsequent network traffic addressed to the device will be delivered to the NCP. To force other device’s ARP cache to update to the new situation, a Gratuitous ARP is sent when the device goes to sleep or wakes up.

D. Communication Protocols

The communication protocols provide a flexible and reliable communication between the NCP service and its client devices. It includes automatic discovery of devices of interest, service registration, action and event registration and notifications etc. We developed the communication module based on HTTP protocol and Multicast DNS (mDNS) for seamless discovery. The mDNS service is based on IP multicast over UDP protocol. The HEG runs mDNS server offering ‘NCP’ service in the local network. This helps the NCP clients to detect the NCP presence and acquire its IP address and port.
number of its HTTP server. After discovery, NCP clients send the registration request along with their state variables to the NCP over HTTP protocol. The HTTP requests are formatted by introducing specific predefined patterns among different data fields which help the HTTP server (running at NCP) to easily identify the received request type and its different fields. The NCP clients monitor the state variables including their power state and immediately inform the NCP if any change occurs. Further, the HTTP-based communication module also allows to register actions and provides automatic event notifications.

VI. EXPERIMENTAL EVALUATION

The main objective here is to check if NCP service works in various realistic scenarios and produces negligible impact on network operations in terms of delay, traffic overhead, packet loss etc.

A. Measurements from the Test-Bed

The tests were conducted under two realistic scenarios to check if NCP can successfully maintain the sleeping client’s presence both inside and outside the LAN:

1) Scenario 1: Third-party host tries to access NCP client from the same LAN.
2) Scenario 2: Third-party host tries to access NCP client from outside the LAN.

The traffic diversion is compulsory in scenario 1 while not required in scenario 2 (as network traffic already passes through HEG). The traffic diversion through Gratuitous ARP on host’s power state transition takes some time during which some packets may be lost. The effect of such delay is roughly evaluated by sending PING messages to the host and counting the lost responses when it switches between sleep and active states. The tests concluded that no PING responses were lost in both scenarios even when sending PING requests at 0.1 second interval. This is because the NCP activates PingReplyAction when it receives sleep notification from its client and deactivates it upon receiving wake-up notification when its client fully recover from sleep mode.

Most applications wait for a limited amount of time when they try to set up a connection; if they do not get a response, the connection is aborted. Sleeping devices should wake up and answer incoming requests before such timeouts occur. The NCP temporary buffers such incoming packets to give enough time to the host for wake up. The latency for waking up a sleeping device includes: (i) the time to send a Wake-On-LAN (WOL) packet, (ii) the time taken by the device to resume from the low-power state and notify the NCP of its new condition. This was evaluated for the WOC action, using the Secure Shell (SSH) remote connection protocol that uses TCP and attempts connection at port number 22. The average values of test results are shown in Table I for both scenarios. The NCP latency is negligible and is measured as time elapsed from the moment of receiving the SSH request to the time when the WOL packet is sent. The SSH response time is evaluated as the time when the third party host sends the SSH request to the time when it receives the response. The host wake-up time represents the time interval between sending WOL packet and the first update packet sent by the host to the NCP.

B. Traffic Overhead

The efficiency of proposed communication module was evaluated by taking into consideration the real data/information, overhead due to data formatting, overhead due to HTTP header and its payload and finally the overhead due to communication paradigm. The real data represents

<table>
<thead>
<tr>
<th>Scenario</th>
<th>SSH Response Time (seconds)</th>
<th>Host Wake-up Time (seconds)</th>
<th>Latency introduced by NCP (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>3.73</td>
<td>6.09</td>
<td>0.12</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>5.29</td>
<td>5.97</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table I: Test results.
Table II: Network overhead during different phases.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Client registration</td>
<td>90</td>
<td>217</td>
<td>755</td>
<td>1519</td>
</tr>
<tr>
<td>Action registration</td>
<td>77</td>
<td>130</td>
<td>662</td>
<td>1426</td>
</tr>
<tr>
<td>State variable update</td>
<td>69</td>
<td>117</td>
<td>653</td>
<td>1351</td>
</tr>
<tr>
<td>Client de-registration</td>
<td>42</td>
<td>85</td>
<td>629</td>
<td>1393</td>
</tr>
</tbody>
</table>

Table III: Expected energy savings.

<table>
<thead>
<tr>
<th></th>
<th>Desktop</th>
<th>Computer Monitors</th>
<th>Laptops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle Wattage (W)</td>
<td>46.2</td>
<td>25.2</td>
<td>14.1</td>
</tr>
<tr>
<td>Sleep Wattage (W)</td>
<td>2.45</td>
<td>0.73</td>
<td>1.44</td>
</tr>
<tr>
<td>Off Wattage (W)</td>
<td>1.47</td>
<td>0.6</td>
<td>0.82</td>
</tr>
<tr>
<td>Avg. idle period/day (hours)</td>
<td>17.5</td>
<td>17.5</td>
<td>17.5</td>
</tr>
<tr>
<td>Full time 'ON' (kWh/year)</td>
<td>404.71</td>
<td>220.75</td>
<td>125.32</td>
</tr>
<tr>
<td>Sleep when idle (kWh/year)</td>
<td>115.76</td>
<td>62.72</td>
<td>37.73</td>
</tr>
<tr>
<td>Full time 'ON' (Euro/year)</td>
<td>89.04</td>
<td>48.56</td>
<td>27.17</td>
</tr>
<tr>
<td>Sleep when idle (Euro/year)</td>
<td>25.47</td>
<td>13.8</td>
<td>8.46</td>
</tr>
<tr>
<td>Savings per device (Euro/year)</td>
<td>63.57</td>
<td>34.76</td>
<td>18.87</td>
</tr>
<tr>
<td>World-wide savings (Billion Euro/year)</td>
<td>51.8</td>
<td>28.33</td>
<td>3.1</td>
</tr>
</tbody>
</table>

the actual size of information in the payload (e.g., IP & MAC addresses, Universally Unique Identifier (UUID), port number, states variables etc). The data was formatted by introducing specific predefined patterns among different data fields before transmitting over HTTP protocol. The communication overhead represents the total size of all the packets exchanged during each individual phase. The analysis were performed during client registration, action registration, state variable update and client de-registration phases. The overhead during each individual phase is presented in Table II. The overhead during client registration is a bit higher compared to other phases as the client sends its complete description along with state variables to the NCP. The messages during other phases only contain information about the client along with specific requests. In short, our HTTP-based communication module produces very small network traffic overhead.

C. Expected Energy Savings

The global energy savings through the adaptation of NCP service depends on the number of its client devices, their active and sleep power and their average use per day. Table III shows the world-wide expected energy savings that are calculated using Energy Star office equipment calculator considering 815 million desktop PCs and monitors and 164 million laptops connected to the Internet. These results assume 22 cent/kWh as average electricity cost in Europe.

D. Implementation Tools

Both, the NCP and its client software were developed using C++ as the main programming language with the support of boost libraries. The mDNS based discovery phase is programmed using AVAHI libraries. Further, the NCP software also used Packet CAPturing (PCAP) libraries in order to sniff packets intended for its sleeping clients.

VII. CONCLUSIONS

This paper has presented the design of a very flexible HEG that can work with heterogeneous devices and communication protocols. It will boost up the development of new services for Smart Grids to provide efficient energy management and better balance of loads with respect to supply capacity. This paper also presented the design of NCP as an example of services that can be offered by the HEG. Further, some preliminary tests were performed that evaluated the correctness of the main software framework and the basic NCP functionalities.

ACKNOWLEDGMENTS

This work was partially supported by the European Commission in the framework of the FP7 ECONET (low Energy CONsumption NETworks) project (contract no. INFSO-ICT-258454).

REFERENCES