

# The Anatomy of a Universal Domotics Integrator for Globally Interconnected Devices

Driart Elshani\*, Pascal Francq

Department of Information and Communication Sciences

Universite Libre de Bruxelles

50, Av. F.D. Roosevelt, CP 123, 1050 Brussels, Belgium

E-mail: delshani@ulb.ac.be, pfrancq@ulb.ac.be

*Abstract—Home Automation Technologies make possible the deployment of home monitoring and control applications. These technologies rely on different networking technologies. Some of them are oriented towards some basic functionalities like switching a device on, off and dimming for lights. Others tend to provide more advanced functionalities like reading of the actual temperature from a thermometer. In this paper, we group them in two categories: low level home automation protocols and high level home automation protocols. We then introduce a universal domotics integrator architecture for a unified home automation system based on the UPnP technology. This architecture could be extended to a large-scale Internet automation groundwork for globally interconnected devices. In this way, inter-device communications and automation procedures could be established via networking automation protocols in conjunction with web automation services. Finally, we study the feasibility, extendibility and inter-operability features of our proposed integrated architecture.*

**Keywords:** home automation, ubiquitous computing, web services, web automation, pervasive computing, integrative systems.

## I. INTRODUCTION

Today, computer networks are already extending their applicability beyond the traditional computer-to-computer communications. The visions for a global network of communicating devices in conjunction with computer networks are being more and more elaborated in the scientific community. It is considered that the next step of computer networks' expansion, will include devices of various natures such as: cars, lamps, home appliance devices, etc., all of them constituting a unified global network of interconnected devices. In this perspective, one can easily draw a parallel between the Internet and the global network of autonomous interconnected devices. This evolving network begins in the home with home automation networks.

Home automation relies on interactions between various devices in the home network. Sometimes these interactions are done with the help of a computer, but nowadays we can notice technologies that tend to provide peer-to-peer communications between connected devices. Home automation may be considered as the first step towards the deployment of a global *ad-hoc* network relying on inter-device communications. Inter-device communications' scope reaches far more beyond our home networks and can be reflected in the deployment of

global telematics services such as location based services, remote control services, remote maintenance services, remote monitoring services, etc.. Moreover, autonomous interconnected devices could operate without a user supervision. This global automated network of (self) communicating devices in conjunction with web automation services, location based services, RFID technology and deployment of IPv6, constitutes the kernel of Next Generation Networks.

This paper is organized as follows. First of all, we survey some home automation technologies. Then, we describe the UPnP [1] technology that tends to act as an integrator with other technologies. We explain how would the UPnP technology serve us as a basement for our integrating system. Finally, we develop on the structure of our integrating system and we study its feasibility, extendibility and inter-operability features.

## II. HOME AUTOMATION PROTOCOLS

Existing Home automation protocols could be grouped in two categories in function of the grouping criteria that we embraced. These grouping criteria are: design, scope and data rate. Consequently, we may distinct: the low level home automation protocols and the high level home automation protocols.

### A. Low Level Home Automation Protocols

These protocols provide limited functionalities like start, stop of the device or execution of a simple and limited set of commands. Normally, a very limited application logic can be added to the end device. To this group of protocols belong: *X10* [2], *Insteon* [3] and *ZigBee* [4]. They provide access to the devices through powerline and/or RF (Radio-Frequency).

### B. High Level Home Automation Protocols

These protocols provide access to the network of connected devices using twisted pair, powerline or RF. Their aim is to build a high speed home networking system. A rich set of commands is attributed to these protocols, thereby enabling them to control devices like a fridge, oven, microwave, washing machine, etc.. All these protocols tend also to behave like integrators in terms of interaction and inter-operability. Today, a significant attention is being paid to: *EIB* [5], *EHS* [6], *LON-Works* [7], *HomePlug* [8], *BACnet* [9] and ultimately *UPnP*. A survey of state-of-the-art home automation technologies is published in [10].

### III. THE UPnP TECHNOLOGY

UPnP is a TCP/IP based, distributed networking architecture which aims at providing connectivity among various networked devices (including PC) in the home, office and everywhere in between. This technology is designed to enable automatic discovery, description and control of devices throughout the network. With UPnP it is possible for a device to join a network dynamically, obtain an IP address, transmit its functionalities and learn about the presence and functionalities of other devices.

Devices must have a full TCP/IP stack built-in in order to support UPnP. The TCP/IP stack is provided by the OS. On top of the TCP/IP stack, UPnP enabled devices possess some embedded services like: a mini web server, an XML (Extensible Markup Language) parser, and a GENA (General Event Notification Architecture). Besides, UPnP compliant devices should support the following protocols: HTTP (HyperText Transfer Protocol), SSDP (Simple Service Discovery Protocol), and SOAP (Simple Object Access Protocol)<sup>1</sup>. All these entities, when combined, form the UPnP architecture, the features of which are presented in the next sections. Before that, let us define some UPnP concepts.

#### A. UPnP Concepts

If a device contains this extended TCP/IP stack, that we shall call a *UPnP stack*, then we say that the device is *UPnP compliant*. Every (UPnP compliant) physical entity (PC, PDA, printer, AV streamer, mobile phone etc.) is (conceptually) a *device* that together with other siblings, form a *UPnP network*. Moreover, every device may contain some *embedded* devices. If this is the case, then the main device that contains the embedded devices is called a *root* device. Each device contains its own *services*.

By definition UPnP is a *peer-to-peer* distributed architecture of (UPnP compliant) networked devices. However, interactions between devices are made in a client-server manner. In fact, any (UPnP compliant) device can play the role of the *Control Point*. Control points can express their interest to monitor and/or control (an)other device(s). In this case, control points can be seen as clients, while devices that they control (which respond to the control point(s) request(s)) can be seen as servers. Indeed, we already know that UPnP compliant devices possess (among other entities) a web server.

#### B. Automatic TCP/IP Device Connectivity

In order to get a network access, devices need to have addresses. Since UPnP is TCP/IP based, then the IP address can be assigned either by DHCP (Dynamic Host Configuration Protocol) or by a mechanism called AutoIP if there is no DHCP server. If there is no DHCP server then the network is said *unmanaged*, otherwise the network is *managed*.

<sup>1</sup>Together, these protocols are often referred to as "UPnP protocols" (in plural).

1) *Managed Networks*: Devices must have a DHCP client in order to search for a DHCP server automatically when the device is connected to the network for the first time. If this is the case, then the device simply obtains an IP address assigned from the DHCP server. Devices are obliged to issue a DHCP request to obtain an IP. Therefore, the AutoIP mechanism is triggered only if a device has not received a response from a DHCP server, which means that the network is unmanaged.

2) *Unmanaged Networks*: In this case, the AutoIP mechanism is enabled. The device must then choose randomly an IP address in the 169.254/16 range, using an implementation dependent algorithm. However, it can happen that the chosen IP address is already in use. Therefore, after picking an IP address the device must test it to verify that the address is not already assigned. In this case, the device picks another address and this procedure is iterated up to an implementation dependent number of retries.

#### C. Automatic Device Discovery

Discovery is the process where devices learn about each others presence. Once a device has an IP address and hence, access to the network, it is the UPnP discovery protocol (SSDP) that allows for the device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices (respectively services) of interest on the network. Basically, in both cases, a discovery message containing some essential specifications about the device (or one of its services) like its type, identifier and a pointer to more detailed information, is exchanged. The UPnP discovery protocol uses multicast for the discovery message exchange. This requires all devices to listen to the standard multicast address for these messages and to respond if any of their embedded devices or services match the search criteria.

SSDP is the protocol that enables the discovery process in a UPnP network. Indeed, this protocol defines the mechanisms for advertising (device (un)available), searching, as well as revoking, of devices or services. It also defines the standard multicast address that will be used. SSDP makes resources available and discoverable through a decentralized communication model based on HTTP. UPnP has a peer-to-peer functionality where each device is aware of all other devices in the network. Hence, there is no central point that keeps track of services and control points. Indeed, control points and services work together to discover each other and make their presence known to the network.

Each time a device (respectively, its service) is connected to the network, it will multicast an *SSDP:alive* message. In this message some information about the service and its location can be found. Control points listen to the multicast address (defined by SSDP) and if they are interested in that particular service, they will add the announcement to their cache. Control points use this cache to keep track of the services they would like to control.

#### D. Automatic Device Description

During the discovery phase, control points retrieve the URL to the device/service's description. This description is constituted by two partitioned, logical parts: the device description and the service description(s). The device description describes the physical and logical containers, whereas service description(s) describe(s) the functionalities exposed by the device.

A single physical device may include multiple logical devices. Logical devices can be presented as a single root device with embedded devices (and services) or as multiple root devices with (no) embedded devices. Therefore, there can exist multiple UPnP device descriptions, one for each embedded device or one UPnP description for the root device (which contains the descriptions for all other embedded devices). In this case, we could talk about an aggregation of devices.

Furthermore, there may exist multiple services for a single device. A printer for example, contains another embedded device, i.e. a fax, which contains two services, i.e. the send fax service and the dial number service. The root device contains three services, i.e. the print service and the other two fax services.

In order to retrieve a UPnP device description, the control point issues an HTTP GET request on the URL provided by the discovery message, after which the corresponding device returns the device description. In the device description, the URL can be found that will be used to retrieve the UPnP service description in a similar process.

The device/service description is written in an XML syntax by the respective vendors. The format of the UPnP device/service description is defined by the UPnP Forum working committee in the form of the UPnP Templates by deriving them from the UPnP Template Language. XML is very suitable for information exchange and can easily be extended to include user (and industry) specified tags. Consequently, in a UPnP network, the details of the device/service description are represented in an XML format, according to the standards defined and coordinated by the UPnP forum committee. Besides, *event messages* are also expressed in XML and formatted using GENA.

#### E. Automatic Device Control

Control points can trigger actions remotely to a UPnP device from which the descriptions of the device and its service(s) have been obtained during the preceding phase. This process is Remote Procedure Call like. Indeed, control points send the action to the device's service and the service returns a result upon action execution. If an action execution does not succeed, then an error is returned from the service. The action is sent by a specific control message to the URL of the service which has been previously provided by the device description.

There are also some state variables which describe the current run-time state of the service. Action execution is reflected by the change of the state variables. Therefore, the effects of an action execution can be seen in the state variables change which are published to the control points by the process of

Eventing.

The core of the UPnP control process is the protocol SOAP. It is a protocol for exchange of information in a decentralized, distributed manner. Indeed, SOAP is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types and a convention for representing remote procedure calls and responses. This protocol makes possible to invoke a distant object (situated at another device) by providing the name of the method and the entry parameters in an XML format via HTTP. The response to the request is also provided in an XML syntax.

Control points may invoke actions on a device service and receive results or errors back. The action, results, and errors are encapsulated in SOAP, sent via HTTP requests, and received via HTTP responses. UPnP uses SOAP to deliver control messages to devices and return results or errors back to control points.

#### F. Automatic Device Eventing

Eventing is a phase where control points listen for state changes at the devices. Indeed, updates from the device's service description(s) are published systematically as soon as there is a change in the state variables. Control points may subscribe to these publications in order to receive updates. In this way, control points monitor the state of the device.

Typically, subscription is done by sending a subscription message and if accepted, the device responds with the duration of the subscription. On the other hand, updates are sent via event messages, which include the names of one (or more) state variable(s) and the actual values of those variables expressed in XML. Messages are delivered via HTTP that has been extended using GENA (General Event Notification Architecture) methods and headers. The HTTP messages are delivered via TCP over IP. These messages can be: *GENA:subscribe*, *GENA:unsubscribe* and *GENA:notify*.

#### G. Automatic Device Presentation

Once the discovery and the description of the device have been obtained, control points can begin with the presentation of the device. This can be done by retrieving the presentation URL situated within the *presentationURL* element in the device description. The page is then loaded to the browser and the user can control the device and/or view the device status.

#### H. UPnP's Performance Issues

We can assert that, if all the devices in an automation network are UPnP compliant, then we would have a functional automation system. This is what the home automation system is about. We would have the possibility to remotely monitor and/or control any device in the home network by simply accessing its URL. Furthermore, devices could interact with each other. Inter-device communications would make possible the automatization of the home. User supervision

would be very small, or even unnecessary. Scenarios like: *if it rains outside then all windows should be closed* would be functionally autonomous because devices would trigger actions between them.

However, UPnP compliant devices are still in the emerging phase and one is forced to have UPnP-like features for devices which don't even have a TCP/IP stack. We would like for example to have the possibility of remote control and/or monitoring of a lamp which understands only on/off and dimming commands. These devices can be controlled through a computer (using X10 for example). Therefore, we have to add UPnP-like features "manually" for non UPnP compliant devices. In the next section, we show that this is possible by creating virtual UPnP logical devices which correspond to the physical non-UPnP-compliant devices<sup>2</sup>.

There are some important questions that arise, concerning the security of the UPnP technology. For example, if we look at the discovery process (SSDP), one might wonder how to limit the broadcast of the exposed UPnP services to a target set of users? We may also wonder how to prevent intruders from a possible interception of a UPnP service transaction? UPnP has been originally designed for SOHO (Small Office, Home Office) networks. Therefore, it becomes vulnerable if the broadcast messages (as well as other UPnP packets) are forwarded beyond the Home IGD (Internet Gateway Device). This means that the home network should be indeed firewalled. On the other hand, the multicast address can be administratively scoped and hence bounded only to the home network devices.

#### IV. A UNIVERSAL DOMOTICS INTEGRATOR

In this section we elaborate a *strategy* for a universal domotics system that theoretically provides integration possibilities for different protocols. This integration system is based on UPnP. Various Low and High Level Home Control Protocols could co-exist within this architecture integrated in a transparent way. Virtually, all High Level Home Control Protocols require devices with some IP connectivity in order to be deployed. Nowadays, some devices already possess full IP connectivity with an embedded TCP/IP stack provided by their respective OS-s. Examples of IP-enabled devices are: network printers, routers, modems, etc.. Other electronic devices do not possess IP connectivity. They are manipulated by Low Level Home Control Protocols and hence, they are connected to the computer via other means like: serial ports, RF, IR (Infra-Red), USB (Universal Serial Bus), etc..

For devices that are IP-enabled, it is easier to monitor (or control) them throughout the home network or web. Indeed, for this kind of devices, the UPnP features can be easily deployed, for example using one of the available SDKs[11]. We shall use the UPnP technology to facilitate our work, which consists in building a generic software integrator for remote monitoring

<sup>2</sup>To anticipate, the UPnP features could be added by the PC acting as a centralized server for the home automation system.

(and/or (self) controlling) of various devices (various in terms of the home automation technology they rely on).

#### V. MOTIVATION FOR AN INTEGRATING SYSTEM

Various networking media present in the house need to be compatible and inter-operable in order to assure a functional and integrated home automation system. In this way, scenarios involving multiple devices in different physical networks, can be made possible (Figure 1).

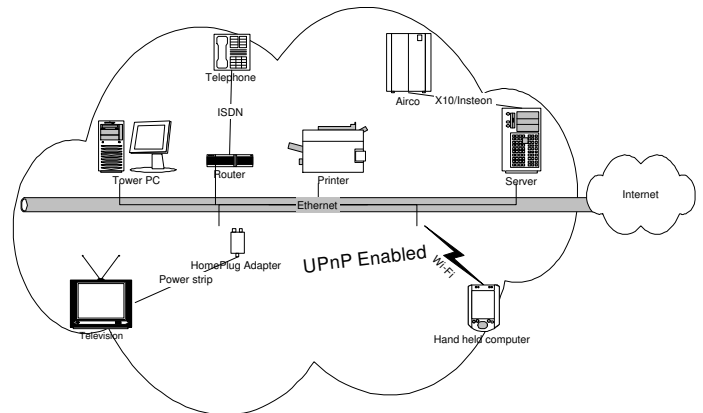


Fig. 1. A possible integration scenario.

#### A. Networks in the Home

System integrators must be aware of as much networking media as possible. This might be difficult to achieve because of the wide range of devices and protocols they rely on. Let us evoke the devices and the corresponding networking technologies, prone to be part of any home automation system.

- White Goods: X10, Insteon, EHS, LONworks, ...
- HVAC BACnet, ...
- Audio/Video: Firewire, Ethernet, Wi-Fi, HomePlug, ...
- Internet Gateways: Ethernet, Wi-Fi, ...
- Telephony: ISDN<sup>3</sup>, DECT<sup>4</sup>
- Electrical Installations: EIB
- Set-top Boxes: Ethernet, HomePlug, Wi-Fi, ...
- Game Consoles: Ethernet, Firewire, HomePlug, Wi-Fi, ...
- PCs: Ethernet, Wi-Fi, HomePlug, ...

Therefore, a common networking protocol is needed to allow (and/or to ease) integration and inter-operability. UPnP serves as an underlying networking medium for the home automation integrated system application that we would like to present.

#### B. Home Automation Scenarios

The home automation integrating system based on UPnP should support different home automation scenarios. We can imagine many of them. Let us give briefly some examples of: wake up scenarios, enter/leave home scenarios and others.

<sup>3</sup>Integrated Services Digital Network.

<sup>4</sup>Digital European Cordless Telecommunication.

### Scenarios: Wake up

- Alarm clock adjustments depending on weather and/or traffic;
- Heating started in bathroom;
- Coffee ready in the kitchen;
- Detection of sleep-over by motion sensors etc..

### Scenarios: Enter/Leave Home

- Locking/Unlocking the main door triggers HomeStatus update event;
- Lighting and/or blinds adjusted;
- Climate (HVAC) changes operating mode;
- Security system is armed/unarmed etc..

### Scenarios: Other

- Lighting: scenes (TV, reading, dinner) configuration;
- Power management: high-energy-consumption devices dependent on tariff;
- Remote control: home automation devices monitoring and/or control (via Web/WAP, speech, gesture);
- Task automation etc..

The number of available scenarios grows exponentially with the number of devices added to the network, assuming that inter-operability features can be provided.

## VI. STRUCTURE OF THE INTEGRATING SYSTEM

Our goal is to deploy a unified Home Automation System based on UPnP, with a PC acting as a Root UPnP Device (i.e. server) with other embedded UPnP devices, which would be able to respond to (command/control or monitoring) requests coming from the Control Points (clients).

We shall use the term UPnP device to describe an entity that has a UPnP Stack (which is a TCP/IP Stack with some advanced functionalities), which means that the device supports the UPnP protocol. This is a logical concept. A UPnP device is a logical device, not necessarily a physical one. Devices that have a UPnP stack built-in physically, will be called UPnP-native. Indeed, it would be interesting to use UPnP for our whole Home Automation System.

First of all, let us reformulate the definition of the UPnP technology. UPnP is an architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices and PCs of all form factors. Actually, it defines general inter-operability mechanisms that enable self-configuring devices to create an ad-hoc, self discovering system of inter-operable network devices by providing us with the means for automatic address configuration, device discovery, command/control, eventing and presentation. We have also seen that UPnP is based on common Internet standards and specifications such as TCP/IP, HTTP and XML.

Furthermore, what is interesting about UPnP is that it is independent of the operating system, programming language and most importantly, of physical network connection. Therefore, we shall use the UPnP technology because we want to build a generic Home Automation System, that means an integrator

for all home control protocols, including X10.

If all devices that belong to the home network were UPnP compliant, then theoretically we could control (or monitor) all of them without additional proceedings. This is because UPnP provides with the necessary mechanisms. However, presently, most devices are not UPnP compliant and they rely on different protocols, often not compatible with each others.

If we use UPnP, we would have a kind of multi tier (or multi layer) architecture so we would need to do only a part of the needed work for a specific home control protocol in order to integrate it (with UPnP) in our Home Automation System, independently of other protocols, without affecting other tiers (layers). We would have a certain hierarchy which would provide us with integration and inter-operability features. Furthermore, we use UPnP to ensure the independence from home device manufacturers.

Let us consider the following scenario: a server PC is on the Home Network and constitutes the Central Command Server for our Home Automation System. In the server we have gateways (programs) UPnP-X10, UPnP-Insteon, UPnP-BACnet, UPnP-Homeplug, etc. that transform incoming UPnP commands to X10, Insteon or other commands. These gateways (programs) operate with entities (objects) that we call UPnP (logical) devices. The server PC responds to Clients (Control Points) situated in the Home Network and/or in the Internet (Figure 2).

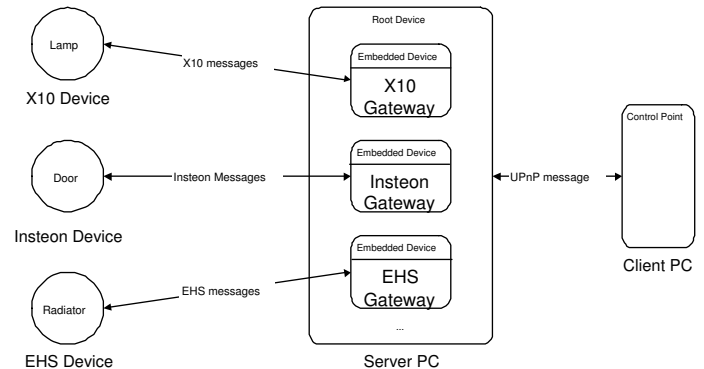


Fig. 2. The general architecture of our integrating system. Rectangles represent UPnP devices. Circles represent physical devices.

## VII. GENERAL ARCHITECTURE

The server PC which is connected to the X10 (and other) devices constituting the Home Network is connected to the Internet and acts as a gateway between the Internet and the Home Control System. Through a (web) application we would control the devices in the house from any place in the Internet. The (web) application would act as a user interface to facilitate the view and the control of a specified device.

However, besides the possibility to remotely control devices in the home network, we would like to have a certain auto-control. This means that we would like to have some automated task execution in one device, in function of the state changes of another device. We have seen that this is possible

in a UPnP enabled network via the eventing process realized using GENA.

For example, if we would like to switch ON the heating in the house in function of the temperature measured by a UPnP compliant thermometer situated somewhere in the network, then the command from the thermometer to the heating controller could be sent automatically without user supervision. In this case the thermometer would be (self) configured as a Control Point interested in the heating controller functions.

If all devices were UPnP compliant, then our Home Automation System would have all the necessary features of interoperability and automatism. Indeed, as we have seen before, UPnP enabled devices can be controlled through control points. We have also seen that UPnP is built on top of TCP/IP, meaning that there is no UPnP without TCP/IP. This means that a device should be TCP/IP enabled, in order to support UPnP. In other words, TCP/IP is a precondition for UPnP. A printer for example is TCP/IP enabled. To make it UPnP enabled, we should add functionalities provided by the UPnP stack layers that are above the TCP/IP layers. However, there are other devices not in the possession of a TCP/IP stack. For this kind of devices, we should add functionalities provided by all UPnP stack layers. Therefore, the various natures of devices raise a problem.

From our point of view, there exist three different types of devices potentially constituting our Home Automation System:

- *UPnP natively enabled devices (UPnP products)*
- *UPnP unable, TCP/IP enabled devices (HomePlug, BACnet products, etc.)* and
- *UPnP, TCP/IP unable devices (X10, Insteon products, etc.).*

In fact, for the first type of devices, it is not necessary to go through a gateway because there is no need for a whatsoever transformation of the UPnP commands coming from the Control Point(s), while for the other two types it is. The idea is to add UPnP capabilities (not provided by the vendors) to these devices, in the Server PC where they are connected to and hence, make them virtually UPnP compliant. Indeed, the TCP/IP connectivity and the UPnP features could be added by developing some gateways (programs) which would transform the UPnP commands coming from the Control Point(s). Therefore, for each Home Control Protocol, we shall have a specific gateway, i.e. UPnP-X10, UPnP-Insteon, UPnP-EIB, UPnP-EHS, UPnP-BACnet, UPnP-Homeplug, etc..

#### A. *UPnP natively enabled devices*

These devices have UPnP built-in. They are "autonomous" in the sense that there is no need to pass through the gateway for them.

Indeed, in order to monitor or control these devices, the UPnP technology suffices, because there is a mini web server built-in in these device types and therefore, all we have to do is to forward the incoming (command) requests to the corresponding device web server, since we know its IP address. Practically, if we have at hand a UPnP natively enabled

washing machine, then there is (among others) a mini web-server on it and if we pass a (command) request, then it is not the Server PC which will respond. The Server PC will (only) forward the request.

#### B. *UPnP unable, TCP/IP enabled devices*

These devices belong to the High Level Home Control Protocols which are based on TCP/IP but are not UPnP enabled. They possess a TCP/IP stack built-in.

Our gateway, in this case, would provide with the necessary functions to assure discovery, description, control, eventing and presentation for this type of devices. We shall elaborate this (a little bit) in the next sections.

#### C. *UPnP, TCP/IP unable devices*

These devices belong to the Low Level Home Control Protocols, which are not UPnP enabled and hence there is no possibility to communicate with these devices, neither to trigger actions on them a posteriori from another device in the home network.

However, the Server PC can communicate through serial ports with these devices. On top of the functions needed to manipulate these devices from the PC, our integrator system should provide with the UPnP related functionalities also, in order to provide device TCP/IP connectivity and other UPnP features. In the next section, we shall give now only some of the possible deployment specifications for these two types of devices.

## VIII. DEPLOYMENT SPECIFICATIONS

Let us elaborate just a little bit on how we could render UPnP compliant these last two categories of devices. Here is an idea on how we could proceed. As we stated earlier, Intel provides a UPnP SDK, that we shall use to build the UPnP devices. These UPnP compliant devices would represent the image of the real physical device. Using Intel UPnP SDK, we could construct any UPnP logical device that we have in mind. This is a feature of Intel UPnP SDK. We specify the device and its corresponding services descriptions, and Intel's UPnP SDK generates the corresponding code. The code generated provides the necessary data structures and functions for a UPnP stack. Besides, there is a generated code, which corresponds to the device's web server. The XML parser is there also. Therefore, we obtain all the services that UPnP provides.

The question that arises now is: we obtain a fully compliant UPnP device with a UPnP stack etc., but what do we need more? In fact, a device that is UPnP compliant can be controlled by other UPnP compliant device(s) and/or can control other UPnP compliant device(s). This is the characteristic of UPnP, by definition, and this indeed is the case in practice. Yet, the messages handled by the UPnP logical device must correspond to the ones handled by the corresponding (real) physical devices. Hence, we must build a gateway (bridge), that will "forward" the incoming UPnP messages to the (real) physical device. Before we do that, we must transform the

UPnP messages to the corresponding Home Control Protocol messages format.

In the case of UPnP and X10 integration (that is the transformation UPnP-message-to-X10-message and vice-versa), we could proceed as follows. On one hand, we have the UPnP APIs provided by the UPnP SDK. On the other hand, we have the Java X10 APIs. Imagine that the code generated by the UPnP SDK is a Java code. Then, the integration is straightforward, because all we have to do is to send an X10 command each time an action is invoked on the corresponding UPnP logical device. To oversimplify, imagine that we have a function called `gateway`. This function accepts as parameters the corresponding state variables of the UPnP logical device and hence, "listens" to the changes in these state variables. If a state variable becomes "on", then the command `X10_Device_On` is invoked. Else, the command `X10_Device_Off` is invoked. One more important detail in this case is the X10 address that corresponds to the physical device. In fact, each UPnP compliant device has a unique identifier. Therefore, we must do a synchronization between the UPnP identifier and the X10 address. This is not a problem, because the X10 address could be the Friendly Name of the UPnP logical device. Indeed, in UPnP, each device has a Friendly Name for identification purposes.

Before we conclude this non exhaustive proposal<sup>5</sup>, let us make one more remark. It is about the messages that go the other way around (from the physical device to a UPnP device). This is not a problem, because the messages go from the UPnP logical device (that corresponds to the (real) physical device). Indeed, if an X10 sensor indicates that a movement is detected, then the corresponding UPnP logical device is aware of this and it would trigger actions on other UPnP devices in function of the movement detection observed from the X10 sensor. We have presented only one possibility on how we could proceed. There are certainly other ways to do this. Yet, as we stated earlier, our goal is only to present our strategy for integration, not to implement it.

## IX. INTEGRATION AND INTER-OPERABILITY ANALYSIS

Our integrator system represents a multi tier architecture composed of: *the UPnP tier*, *the gateway tier* and *the home control protocol tier*. Each of these tiers is independent of each others. Therefore, a specific Home Control Protocol could be easily added, without affecting the other tiers.

In general, a good integrator system should provide us with the possibilities of integration between all home purpose electronic devices. This means that the more Home Control Protocols are supported, the better is. If we have a communication, we have a complete automation, in the sense that devices would trigger the functionalities between them independently, without user supervision.

Passing from one Home Control Protocol to another should be transparent. This means that Control Points ignore the fact that there is a transformation of UPnP commands into another

Home Control Protocol commands. Controlling an X10 device should be done in the same time, potentially, as controlling a device which uses another Home Control Protocol. The user would simply add a device to the home network and thus have a full integration.

Different Home Control Protocols coexist with each others in this integrator scheme, because:

- *they use different network media (serial port, IP, ...);*
- *they operate with different frequencies in the powerline;*
- *they use (physical) devices specifically built for that Home Control Protocol.*

Therefore, we obtain a fully integrated, inter-operable, extendible and compatible system for our unified and centralized home automation system. This unified system could be extended beyond the home network, and reach other home automation networks. Integrated home automation systems with globally interconnected devices could be deployed.

## X. CONCLUSION

With rapid technology evolutions, networks are becoming faster and more reliable, thereby making possible the deployment of the Next Generation Automated Networks (NGAN). Home automation systems will be an integral part of a global NGAN. In this evolving global automation network, every device, even every application, is likely to have an address. Secure end-to-end communications should be supported. Intelligent environments should be easily deployed. All of this should be integrated through an extended, scalable and robust Internet connectivity. These environments should be (remotely) monitored and/or controlled even without user supervision. This should be made possible through inter-device communications and thereby, through the deployment of ad-hoc networks.

In this paper, we have presented an integrative domotics architecture, that could be extended beyond the home network, and reach other home automation networks. We can already claim that NGAN is becoming a matter of fact in our life and that tomorrow's next generation Internet has already started to be deployed, today.

## REFERENCES

- [1] Universal Plug and Play (UPnP) Technical Whitepaper-<http://www.upnp.org/>, 2006
- [2] X10 Technical Whitepaper-<http://www.x10.com/>, 2006
- [3] Insteon Technical Whitepaper-<http://www.insteon.net/>, 2006
- [4] ZigBee Technical Whitepaper-<http://www.zigbee.org/>, 2006
- [5] European Installation Bus-<http://www.eiba.be/>, 2005
- [6] European Home Systems-<http://www.ehsa.be/>, 2005
- [7] Hoske, M.T., "Lonworks expands, positions as universal network solution", *Control Engineering*. Vol. 43, no. 7, pp. 77-78. 1996
- [8] M. K. Lee, R. E. Newman, H. A. Latchman, S. Katar, L. Yonge, "HomePlug 1.0 powerline communication LANs - protocol description and performance results", Volume 16, Issue 5 , p.447 - 473, 2003
- [9] Haakenstad, L.K., "The open protocol standard for computerized building systems:BACnet", *Proceedings of the 1999 IEEE International Conference on Control Applications*, Volume 2, p.1585-1590, 1999
- [10] Driart Elshani, "A Survey of Existing Home Automation Technologies: Integration Issues", Master Thesis, June 2005, Universite Libre de Bruxelles
- [11] Intel's UPnP SDK-<http://www.intel.com/technology/UPnP/download.htm>

<sup>5</sup>For other protocols, we could proceed in a similar way.