Available online at www.sciencedirect.com
ScienceDirect
Journal of Network and Computer Applications 30 (2007) 499–514
Journal of NETWORK and COMPUTER APPLICATIONS
www.elsevier.com/locate/jnca

Embedded web server-based home appliance networks

M. Can Filibeli, Oznur Ozkasap*, M. Reha Civanlar

Department of Computer Engineering, Koc University, Rumeli Feneri Yolu, Sariyer 34450, Istanbul, Turkey

Received 18 July 2005; received in revised form 29 March 2006; accepted 13 April 2006

Abstract

Powerful microcontrollers are used as parts of most home and office appliances of today. Integrating web servers to these intelligent devices will aid in controlling them over the Internet and also in creating effective user interfaces in the form of web pages. Assigning multiple functionalities to a single button on an appliance help manufacturers economize user interfaces, but, this can easily create confusion for the users. Since the cost of web-based interfaces is considerably low, they can be used to provide the infrastructure for the design of simple and more user-friendly interfaces for household appliances. Also, a web page based interface is much easier to change, when needed, as compared to a hardware interface. This paper presents a novel approach to control devices with embedded web servers over the Internet and to form device networks such that their components can make use of one another's services and functions while improving the user interfaces. The main benefits of this approach include its lightweight design, automatic configuration, and, utilization of widely available and tested network protocols of TCP/IP and HTTP. The validity of the approach has been verified through a prototype system working with real appliances.

Keywords: Embedded web servers; Home network systems; Ethernut; Home appliance networks; Device discovery

1. Introduction

Many home and office appliances such as televisions, VCRs, telephones, watches, ovens, toasters, dish-washers and thermostats produced over the last decade are equipped with

*Corresponding author. Tel.: +90 212 338 1584; fax: +90 212 338 1548.
E-mail address: oozkasap@ku.edu.tr (O. Ozkasap).

1084-8045/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2006.04.001

complex features. This has become possible by the computer industry's ability to fit more transistors into a smaller area of silicon which increases the computational capabilities of devices while decreasing the cost and the power consumption. Nowadays, most of the home and office devices contain embedded microcontrollers. One of the most important reasons for embedding computers to everyday appliances is that they make the appliance easy to use while giving them new features that enhance their values (Borriello and Want, 2000). As of today, these embedded computers cost a few dollars and are as capable as desktop computers of 1980s. Such devices can support an embedded operating system with a TCP/IP stack, interface to a 10/100 Mbps network and can be used as web servers.

The Internet has been mostly used to connect personal computers so far, but shortly all kinds of appliances with embedded computers will exchange information over the Internet. A massive number of microcontrollers are available in today's devices which can be linked to the Internet. If these intelligent appliances could be connected to the Internet at low cost, the way we control and manage their functions would change entirely.

As the abilities of appliances increase, the complexity of controlling these devices increases as well. The need for informative user interfaces is suppressed by the higher cost of implementation. Therefore, complex appliances come with poorly labeled buttons and overloaded functions. For instance, to set the timer of a stereo, a user should press a combination of buttons within a few seconds. To ease the interaction with complex appliances, the scenario that every device has a keyboard and a display is not an option considering an environment with thousands of embedded systems. We clearly should come up with a compact solution. Web pages can be used as user interfaces where each appliance has the ability to serve its user interface as a web page over the Internet. These pages can contain not only text but also pictures, hypertexts, photographs, video and audio as in usual web pages. By the advent of the user interfaces based on web pages, all devices requiring user interaction can be controlled and managed from one device which includes a web browser, such as a personal digital assistant, cell phone, etc. Also, use of web page based, button and display free designs reduce the cost of production while making the systems more user-friendly.

Remote control via the Internet is not a new feature and used in home automation systems. However, providing a mechanism for interaction between devices in this environment is quite challenging. For example, an alarm clock can interact with the heating system according to the alarm time and can send instructions to the heating system to start warming the house before the residents wake up. There exist some commercial platforms, which facilitate the development of applications interacting with and exploiting the abilities of networked intelligent devices, such as Universal Plug and Play (UPnP) (UPnP Forum; Jini Community; Pascoe, 1999) etc. Our system is not similar to these platforms. However, all solutions including ours aim at the same features, namely supporting remote control of devices via the Internet and providing a mechanism for interaction between devices. We believe that an attractive alternative to the existing techniques is a web servers based system, where each appliance runs an embedded web server serving the pages needed for its own user interface. This approach is considerably simpler than the existing systems while keeping most of their functionalities as will be explained in this paper.

We designed an application architecture, and, built and demonstrated a prototype system that can integrate web servers to everyday appliances with embedded microcontrollers to control and manage them via web pages using regular web browsers

over the Internet. The system allows forming device networks such that their components can easily make use of each others' services and functions. The unique architecture presented in this paper is used to develop the prototype system which, in turn, used to test and demonstrate the characteristics of our design. The main contributions of this architecture are its lightweight design, automatic configuration and utilization of widely available network protocols of TCP/IP and HTTP.

The remainder of this paper is organized as follows: Section 2 discusses everyday appliances with web servers. It details out the Ethernut platform and Ethernut Web Server, the basic components that we used in building our platform. Section 3 presents the device coordination in home and office networks. Platforms such as UPnP, Jini, and Salutation are introduced and compared. In Section 4, Ethernut based architecture and our prototype system design is introduced and compared with other available platforms. Section 5 discusses the benefits and drawbacks of the Home Appliance Network architecture. Finally, Section 6 presents our conclusions.

2. Everyday appliances with web servers

Recent technological developments enable us to embed web servers into everyday appliances with low costs. What do we hope to gain by putting our devices online? Basically, the web enables users to fetch web pages and display them on their own browsers in a platform-independent manner. Moreover, information coming from an appliance's sensors can be used to generate dynamic HTML documents by common gateway interface (CGI) scripts. Therefore, any device connected this way can be monitored and controlled through CGI scripts and the results can be sent to the user's browser as a web page. Not only users, but also manufacturers can use web access to update their products' software simply by uploading the new software to devices remotely or appliances can be programmed to regularly check the manufacturer's web site for software updates and download them automatically when they are available. Furthermore, when an appliance requires maintenance, it can automatically inform the technical staff without interrupting the user.

Web-based interfaces are cross-platform and easier to develop. The costs related to product documentation, training and support are lower. These can be instrumental in solving user interface related problems. Most complex appliances have many unused functions, because users find them too difficult to use. There are several reasons for poor interfaces. An important one of these is that since manufacturers economize on buttons and displays, complex features can be used only by button combinations with no or very hard to understand indications on the button labels. Moreover, the feedback given to the user after a performed action is usually not satisfactory (Nichols and Myers, 2003). On the other hand, as manifested by the exponential growth of the number of web users globally, ordinary people can use web based interfaces effortlessly without any knowledge of the underlying hardware and software.

There are many industrial and academic groups that are working on generating interfaces for complex appliances (Jini Community; HAVi; Nichols et al., 2002). Human Computer Interaction Institute of Carnegie Mellon University investigates how handheld devices can be used to improve the interfaces for everyday appliances, using an approach called personal universal controller (PUC). PUC exchanges information with the appliance. First, it downloads the description of the appliance's functions, then it creates a control panel using transferred description and finally it sends control signals to the

appliance as the user operates the panel. The control panel contains figures, informative texts and extra features that ease the usage of the appliance. The research is focused on automatic generation of control panels. Two studies have been conducted in order to create high quality and user friendly control panels. The first study is a between-subjects comparison of a handheld interface with the interface to an actual device (Ponnekanti et al., 2001). Paper prototypes (Rettig, 1994) were used in the first study as the handheld interface. In the second study, actual implementations of handheld interfaces were used instead of paper prototypes. Both studies showed that users were able to complete complex tasks in about one half of the time and with half as many errors as the actual device interfaces (Nichols et al., 2002). The results of experiments support the need for more informative user interfaces. In the sense of simplifying the user interface of the appliance, the handheld interface mentioned above and the web-based interface presented in this paper both address the same requirements. Therefore, users can control complex appliances easily with well designed, feature rich web-based interfaces which contain self-informative buttons, hyperlinks, texts, figures, etc.

At this point, we should indicate that providing user interfaces as web pages by no means guarantee having well designed user interfaces. However, they facilitate easy design and maintenance of well designed user interfaces based on web pages and associated design tools. This paper focuses on the infrastructures for using web pages as user interfaces. Web page design for better user interfaces is outside our scope.

The benefits of embedding a web server into an appliance can be summarized as follows:

- Users can interact with appliances using a device which contains a web browser. Such devices, ranging from PCs to cell phones, are universally available and using them is common knowledge.
- User-interface hardware (electromechanical) costs can be eliminated and more user-friendly interfaces can be designed with low costs.
- Controlling, monitoring and updating can be done from anywhere in the world.
- Products can be supported throughout their life cycles by manufacturers by uploading new software versions remotely, reducing the maintenance costs.
- User interfaces can be updated or extended. For example, accessibility options for helping handicapped users can be added. User interface can be featured in different languages, etc.

2.1. Ethernut—open source project for building ethernet devices

An embedded web server should use the HTTP protocol to transmit Web pages from the embedded system to the web browser and to transmit form data back to the embedded system attached to the appliance. The embedded system requires a network interface, such as Ethernet, a TCP/IP protocol stack, embedded web server software and static and dynamic web pages that form the user interface for that specific device. Because the embedded systems have limited CPU and memory resources and these resources are mostly used by real-time applications, end-users may have to wait up to few seconds for an HTTP response. Multi-threading should be employed in the embedded systems to avoid slow response. Moreover, web server software must be able to respond to multiple HTTP requests simultaneously and must use non-blocking multithreading such that one HTTP transaction waiting for data will not prevent another HTTP transaction.

When implementing the TCP/IP stack for microcontrollers, some features can be omitted depending on the specific application requirements. In order to serve wide range of application requirements while keeping the size compact, an embedded system should implement five basic protocols of the TCP/IP protocol suite: TCP, UDP, IP, ICMP and ARP. Moreover, application level protocols DHCP, DNS and HTTP should also be implemented.

Users can interact with appliances by submitting a form to an embedded web server. The form data is processed by a CGI script and the results are returned back to the user as an HTML page. Appliances can be monitored and controlled by forms.

Hardware of a web-enabled embedded system must have a powerful and low-cost microcontroller, compact size, flash/EEPROM for accommodating the TCP/IP stack and an Ethernet controller.

Taking all these software and hardware parameters into consideration, we chose the Ethernut system, which is an open source hardware and software project for building embedded Ethernet devices, as the core component of our web accessible home appliance network system.

The Ethernut system includes a small board (80 mm × 100 mm), which is equipped with Atmel Atmega 128 CPU running at 14.7456 MHz with 4 KB internal SRAM, full duplex IEEE 802.2 compliant Realtek 10 Mbps TRL8019AS (Ethernut 1) or 100 Mbps LAN91C111 (Ethernut 2) Ethernet controller, 2 RS-232 serial ports, on board RJ-45 and DB9 connector, 22 programmable digital I/O, 8 channel, 10 bit analog/digital converter, 2 × 8 bit and 2 × 16 bit timers/counters. The Ethernut 1 and Ethernut 2 boards are equipped with 32 and 480 KB external SRAM, respectively.

The software part of the project consists of a real time operating system (Nut/OS), which supports dynamic memory management, event queues, stream I/O functions, simple Flash ROM file system, and a TCP/IP protocol suite (Nut/Net) which includes ARP, IP, ICMP, TCP DCHP, HTTP API with file system access and CGI functions. The source code is well-documented and implemented mostly in C. Wide range of applications were developed using the Ethernut (Ethernut Hardware Manual) such as networked sensors, remote monitoring equipment, alarm service providing, remote diagnose and service, industrial Ethernet applications home/building control.

Web servers running on normal computers with mass storage devices use the file system to retrieve the requested web document. To offer similar functionality, Ethernut contains a very simple file system which is based on data structures stored in the flash memory. The available space is very limited in embedded systems to store web pages; however, web pages require only flash ROM space and SRAM is not utilized (with typical applications at least 64 KB of web page storage is available).

One of the ways of interfacing applications with web servers is to employ CGI. If a form is submitted by the user, the web server executes the related CGI script and the resulting web page is dynamically formed and returned to the user. By this way, user can monitor the appliance's status or control it through the digital I/O ports.

3. Device coordination in home and office networks

Letting the home devices communicate, interact and share functions/services with each other requires some sort of device coordination within the appliance network such that devices can enter the network, announce their services and find services offered by other

devices. For enabling device coordination, a subset of the following capabilities should be provided to a device (Rekesh, 1999):

- declare its existence to the appliance network,
- discover the appliances in the network automatically,
- describe its services and capabilities to other devices,
- query and understand the capabilities of other devices,
- configure itself automatically,
- inter-operate with other devices wherever meaningful.

Some of the best-known commercial platforms for facilitating the development of distributed applications interacting with and exploiting the capabilities of diverse networked appliances is Jini, UPnP, Salutation and HAVi. Next, we describe related work including these commercial platforms as well as relevant research activities.

3.1. Commercial platforms

Jini from Sun Microsystems is based on Java technology. It is a programming model and infrastructure that aims to enable spontaneously networked devices to communicate, interact, and share each others' services. The key idea of Jini is to allow any device with a processor, memory and network connection to provide services to other entities in a network and to use such services offered by others. Jini provides mechanisms to enable devices to plug together to form a Federation. Components of the federation may be physical devices or software objects written in Java. In Jini terminology, such devices and objects are called Services which can be connected to a network and announce their presence to a Lookup Service. Clients that wish to use these Services locate and call them to perform tasks.

The Jini system contains three main types of players which are connected to a network. There is a Service, such as a VCR, a printer, a digital camera, a coffee machine, etc., a Client that would like to use this Service and a Lookup Service that acts as an agent between services and Clients. Jini is based on Java technology, so it exploits several advantages of Java. It turns a heterogeneous device network into a homogeneous Java Virtual Machines network, which supports the Write-Once-Run-Anywhere capability and operating system independence. On the other hand, a device plugged into the network has to be assigned with an IP address and a subnet mask. Thus, administration is still needed for the network. The most important disadvantage of the Jini System is that running Java still costs a lot of processing power. If you want to run JVM on limited resources, such as on an 8-bit microcontroller, the only choice has been the reduced subset of Java created specially for the environment. Depending on processor, application, and component set, a typical core code needs a memory space of 40 kbytes (Bettstetter and Renner, 2000).

UPnP from Microsoft is an architecture defined at a much lower level than Jini. Instead of using virtual machines to change heterogeneous device networks to homogeneous ones, protocols such as TCP/IP and standards like HTTP and XML which can easily be implemented on all devices are used.

UPnP has five major components: Device Discovery, Device Description, Presentation, Events and Subscriptions and Control. Device discovery enables devices to announce their presence to the network as well as discover other available devices in the network. During

discovery, the information that is exchanged, provide basic information about the devices and their services, and a description URL, which can be used to gather additional information about the device. A device may contain a set of services corresponding to each functional component of the device. UPnP keeps description of these services in XML files. In the device discovery phase, minimum information about the capabilities of that particular device is transferred with the discovery messages. In the description phase, control point learns more about the device by retrieving the XML file from the URL provided by the device discovery message. If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a browser and depending on the capabilities provided, a user can control the device and/or view its status. When a control point subscribes to a service, the service sends event messages to the control point to announce changes in device status. Control points use URLs that are provided during the description process to access additional XML information that describes actions to which the UPnP device services respond, with parameters for each action. Control messages are formatted in XML and use SOAP.

A device plugged into the network can receive its IP address and subnet mask either by dynamic host configuration protocol (DHCP) or AutoIP (Rekesh, 1999). If there is a DHCP server in the network, device is configured automatically by DHCP. On the other hand in AutoIP mechanism, the device randomly chooses an IP address from a reserved range and then sends out a request with address resolution protocol (ARP) to detect whether the IP address is used by another device. UPnP does not specify any security mechanism for access and control of UPnP devices. Security has to be provided by the applications or services. Security standards on lower networking layers such as SSL may be used to gain security. The security features of this architecture have not been developed yet.

Salutation, developed by the Salutation Consortium, is cooperation architecture to solve the problems of service discovery and utilization among diverse appliances and equipment. The Salutation architecture provides a standard method for devices to describe and advertise their capabilities to possible Clients. It also enables devices to search other applications, services and devices for a particular capability, and to request and establish sessions with them to utilize the capability (Salutation Consortium, 1999a,; Salutation Consortium, 1999b). "Salutation is shipping while others are still thinking!" claims the Salutation Consortium in (Pascoe, 1999) while comparing its solution to Jini and UPnP. Products equipped with Salutation are available in the market since 1997. Six member companies (Canon, Fuji, IBM Japan, Mita, Muratec and Ricoh) demonstrated some Salutation-enabled products in 1997. These products support a company intranet that automatically detects new peripherals as they get attached and makes their capabilities available for use. The new products include both hardware and software applications. Demonstrated products are digital copiers, fax machines, scanners and printers. Many of them integrate with Lotus Notes in a corporate intranet environment.

Salutation is nonproprietary and specifications are available for third-party development. It is also a relatively lightweight platform, which is a significant benefit for current mobile phones and PDAs. Salutation is not suitable for dynamic and large-scale networks. Currently, its specification is being modified to better suit large-scale networks. Unlike UPnP, Salutation does not address the automatic configuration because of its transport independent design. Moreover, it would not be easy to propose a solution to this problem that worked on all transport types. Thus, its use for home networking is not easy, since it

needs administration for configuration. Moreover, Salutation does not specify any security mechanism for access and control of networked devices. Therefore, security mechanism has to be provided by the applications or services.

Home Audio/Video Interoperability (HAVi) is a consortium of consumer electronics companies organized to define standards for interoperability among network connected digital home entertainment products. HAVi compliant devices automatically announce their presence and capabilities to every other device on the HAVi network. Architecture is designed to work over an IEEE 1394 network. No configuration of network addresses or device drivers are needed in the HAVi system.

The device discovery architectures given above, approach the problem of service discovery very well. Jini and UPnP are both flexible and scalable. Although Salutation is not suitable for dynamic and large-scale networks, it addresses mobile devices successfully because of its lightweight design. Moreover, it is the only platform used in commercial products in today's market.

Jini is based on dynamically downloadable Java code which is used to tie various devices together, but this structure makes Jini to employ Java technology in the development. On the other hand, UPnP is based on commonly used protocols and standards which can be implemented easily on every device using any technology. Moreover, UPnP is the only solution that supports automatic configuration of devices using either DHCP or AutoIP. Salutation is well defined but confined to the service discovery protocol and session management. It can operate in any network, including IP, IR and wireless. This transport independence is the strongest feature of Salutation. Salutation doesn't address features like remote event notifications. HAVi is designed to enable interoperability only in home AV networks and supports multimedia streaming and service reservation. HAVi architecture is not suitable to be used with all home appliances but only with multimedia ones.

If these frameworks are analyzed by comparing the features each one provides, Jini is the most sophisticated, providing device discovery, device lookup, leasing, service invocation, and distributed events. UPnP targets a lower level by supporting device discovery and network configuration. Salutation and HAVi target a middle level, enabling discovery and service invocation.

A number of research groups are working on the design and implementation of home network systems using middleware technologies described above (Dong-Sung Kim et al., 2002; Nakajima, 2002). However, none of the above four has wide scale deployment. UPnP's self-configuration feature is an essential mechanism that should be implemented on other platforms. Many more changes and additions seem to be needed before having a complete and standardized device connection platform that is commercially usable.

3.2. Relevant research activities

A number of research groups are working on controlling appliances from handheld devices. The Stanford ICrafter (Ponnekanti et al., 2001) is a framework for distributing appliance interfaces to several controlling devices. Their structure supports the automatic generation of interfaces. The XWeb project (Olsen Jr et al., 2000) aims to separate functionality of the appliance from the device upon which it is displayed. XWeb defines an XML language from which user interfaces can be created. The PUC is another framework for improving interfaces to complex devices by introducing an intermediary interface. A PUC engages in two-way communication with everyday appliances, first downloading a

description of the appliance's functions, and then automatically creating an interface for controlling the appliance.

The research activities mentioned above focus on automatic creation of user interfaces rather than the system and infrastructure issues. These frameworks separate the appliance from the interface. Interfaces are generated using specification languages depending on the type of the control device.

4. Ethernut-based embedded web servers

Device networks based on Jini, UPnP and Salutation platforms need specialized servers to be accessed over the Internet. On the other hand, we can connect our home appliance networks together and enable access to appliances over the Internet using the Ethernut. The architecture overview is given below and illustrated in Fig. 1.

4.1. Architecture

The *Home Appliance Network* (HAN) connects to the Internet through the *Ethernut Home Server* (EHS). The Internet connection can be established either via an analog modem over PSTN or via a GSM modem over GPRS using the point-to-point protocol (PPP) (RFC 1661). Appliances within a HAN can connect to the network either by wired (10/100baseT) or wireless (IEEE 802.11) network technologies. After connecting to the Internet, EHS sends its IP address and *Unique Home Identifier* (UHI) to an *Address Server*. Address Server keeps UHI, IP address couple for a certain period unless it is renewed. This facilitates robust behavior in the case of Internet connection failures and IP address changes, as the Address Server will not contain HANs that are no longer available. Moreover, the Address Server prevents unauthorized access to the HANs. An Address Server with a static IP is necessary in this architecture since in every PPP connection EHS

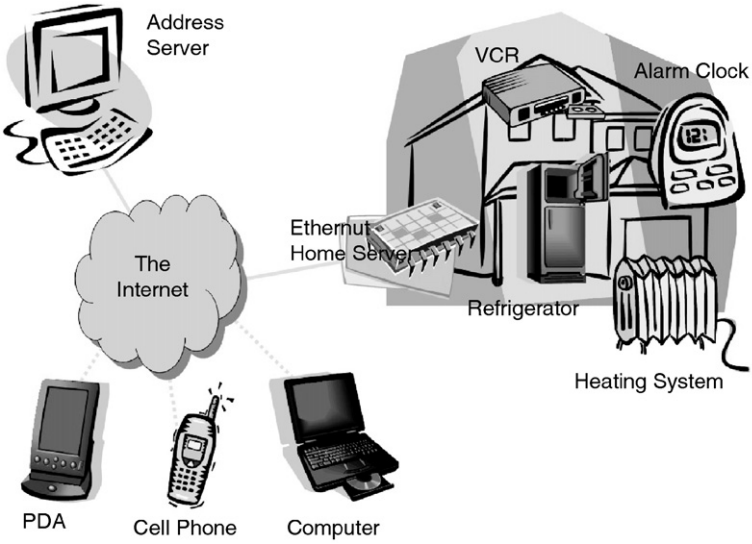


Fig. 1. Ethernut-based embedded web servers architecture.

may get a different IP address which must be known by the user in order to reach the correct HAN over the Internet.

When an appliance is plugged into the network, it obtains its IP address, subnet mask and gateway address from a DHCP server if it exists in the network. If no DHCP server is available, appliance randomly chooses an IP address from a reserved range of IP addresses and then sends out a request using ARP to detect whether the IP address is used by another device. If the address is used by another device, self-configuration procedure is started over with a new IP address chosen randomly from the specified range.

After self-configuration finishes, the appliance locates the EHS by broadcast unless it knows the EHS's IP address. If the EHS's address is known by the appliance, unicast can be used to send the information. The appliance sends its identifier, IP and MAC address to the EHS. The identifier contains a descriptive name of the appliance, understandable by a user, such as "The Internet Clock". If an EHS is not available in the network, appliance starts serving its web pages within the network and waits for an *EHS online announcement*. When an EHS becomes online in the network, it broadcasts an online announcement, so that appliances that are already in the network register themselves to the EHS.

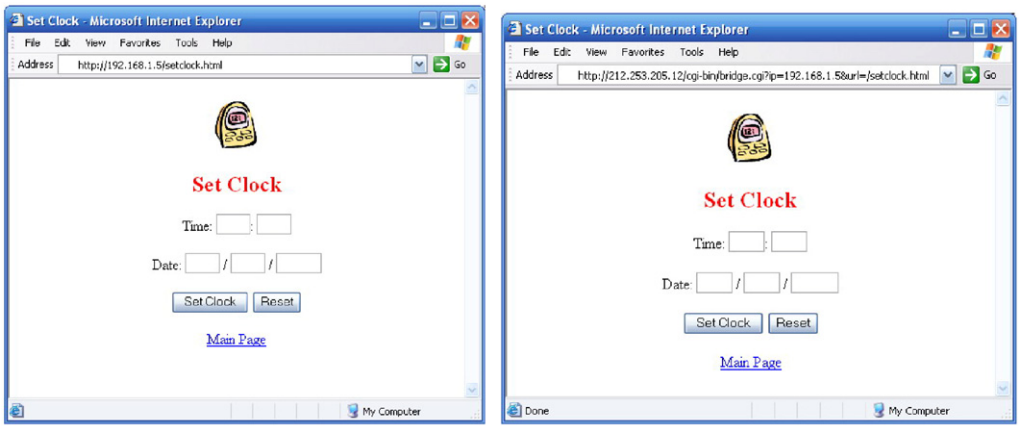
Each appliance within a HAN includes a web server that keeps web pages to control and monitor its functions and status. Services of an appliance can be executed using the hyperlinks to CGI functions on this web server. A service request to a device within the HAN coming via the Internet is routed to the target device by the EHS.

After the EHS gets the information from an appliance, it uses the identifier and the IP Address to construct a hyperlink on its main web page. The address contains a link to a CGI function on the EHS with parameters: an IP address, an MAC address, an Identifier and a URL which points to the root of the appliance's web server. This data embedding into the web page is needed because one EHS is used to control several appliances and the web servers are stateless. When a user clicks on one of the hyperlinks in the main page of the EHS, the bridge CGI function running on the EHS opens a TCP socket to the target appliance and receives appliance's main web page through this socket. Fig. 2(a) demonstrates a sample web page for an appliance.

Before sending the web page to the requester, CGI function modifies the source of the web page such that every hyperlink, form target, and image points to the same CGI function (bridge.cgi) on the EHS (the IP Address of the EHS assigned by the PPP connection is 212.253.205.12) with a different URL parameter containing the original link on the appliance's web server. The modified web page is given in Fig. 2(b). For example, the original hyperlink (〈a href = "/index.html"〉Main Page〈/a〉) is modified by the EHS before sending to the requester. The hyperlink is equipped with the IP address of the EHS's PPP connection and the IP address of the target appliance:

〈a href = "http://212.253.205.12/cgi-bin/bridge.cgi?ip = 192.168.1.5&url = /index.html"〉 Main Page〈/a〉

If the requester clicks one of these hyperlinks, again the CGI function opens a TCP socket and performs actions on the target device (the IP address of the target device is 192.168.1.5) according to the URL parameters and retrieves the resulting web page. Once more, it modifies the web page so that later requests to the appliance are directed over the EHS. A hyperlink pointing to an appliance exists on the main web page of the EHS for a certain time unless the information is renewed. This leasing mechanism is crucial for the robust behavior of the system in case of appliance connection failures.



(a) (b)

Fig. 2. Bridge.cgi Function—(a) Browser view of the original clock setting page. (b) Browser view of the modified clock setting page.

The architecture also supports device coordination. For instance, an alarm clock may interact with the heating system according to the alarm time and send instructions to the heating system. An appliance sends instructions to another appliance by requesting the CGI function of the pre-specified service (for instance: start warming the house) with parameters.

The architecture contains two single-points of failure: the address server and the EHS. In order to avoid these, a secondary address server and a secondary ethernut home server can easily be integrated to the system.

4.2. *The prototype implementation*

We have created a prototype for demonstrating and testing our design. Fig. 3 illustrates an overview of the prototype system. We implemented an Internet alarm clock (Fig. 4a) and an Internet TV remote controller (Fig. 4b) to be used as web embedded appliance samples controlled within our prototype.

The Internet alarm clock is implemented as a small board which houses an Atmel ATMega16 microprocessor and an LCD to display the time and date. The clock is designed without a dedicated user interface (without buttons for setting the date and time). Instead, an Ethernut board is connected to the clock. The Ethernut board contains a web server that serves web pages that constituting the user interface of the clock. The Internet TV remote controller consists of another board containing an Atmel ATMega16 microcontroller and an IR LED transmitter. Unlike the conventional remote controllers, this Internet TV remote controller has no buttons on it! Users interact with it via web pages that are served by an Ethernut board which is connected the remote controller.

In the prototype, the alarm clock acts as a central, single-point controller for the user. Instead of programming the heater, coffee maker, bread grill, water heater, one by one, the user programs just the alarm clock. Moreover, the web interface makes programming appliances very easy via the alarm clock. Considering the cost of the embedded system, we think it worth to have an embedded system in the alarm clock.

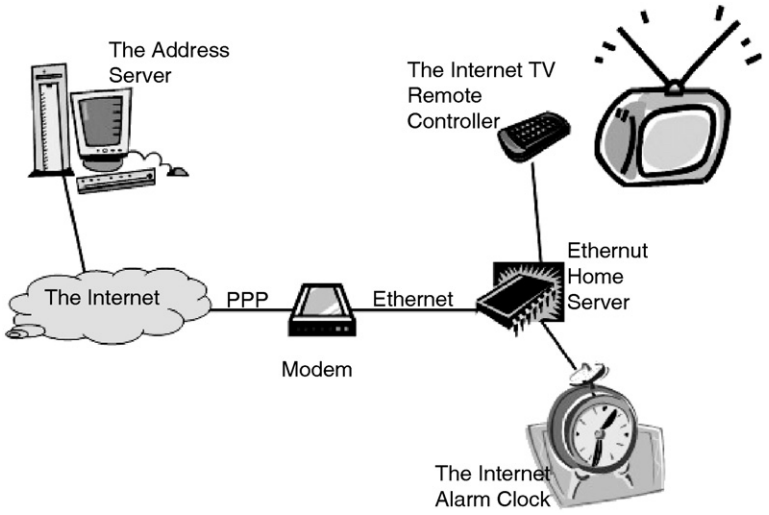


Fig. 3. The prototype system overview.

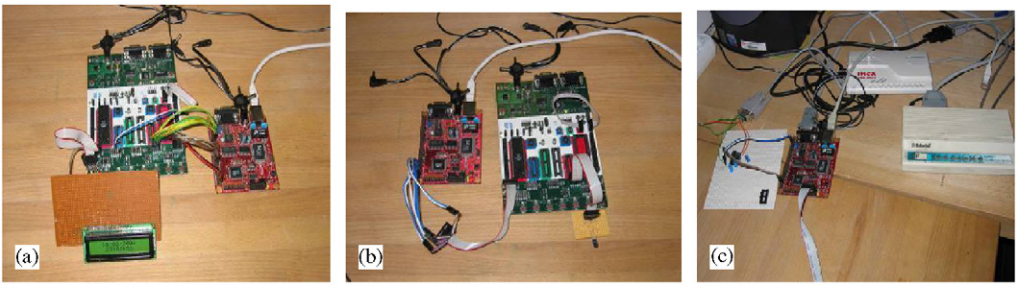


Fig. 4. The Hardware of the Prototype ımplementation: (a) Alarm clock with embedded web server. (b) VCR remote controller with embedded web server. (c) The Ethernut home server.

Two appliances are connected to an Ethernet via Ethernut boards. Automatic configuration property of the system was tested on the two networks, one of them housing a DHCP server. We employ an analog modem and it is connected to the EHS via RS232 (Fig. 4c). EHS connects to the Internet using the PPP. After the connection, EHS registers itself to the address server with its IP address. The alarm clock and the remote controller register themselves to the EHS as soon as they are plugged into the Ethernet.

Users can get the IP address of the EHS from the address server. When a user connects to the EHS using a web browser, a webpage showing the hyperlinks of the online appliances (in our prototype, the TV remote controller and the alarm clock) are displayed as shown in Fig. 5a. Via this structure, the user can set the alarm of the clock (Fig. 5b), and at the alarm time a selected remote controller initiates a selected command on a chosen appliance as shown in Fig. 5c and d.

In our prototype, in addition to providing device control over the Internet, the coordination between the appliances in the HAN is implemented as well. Users without

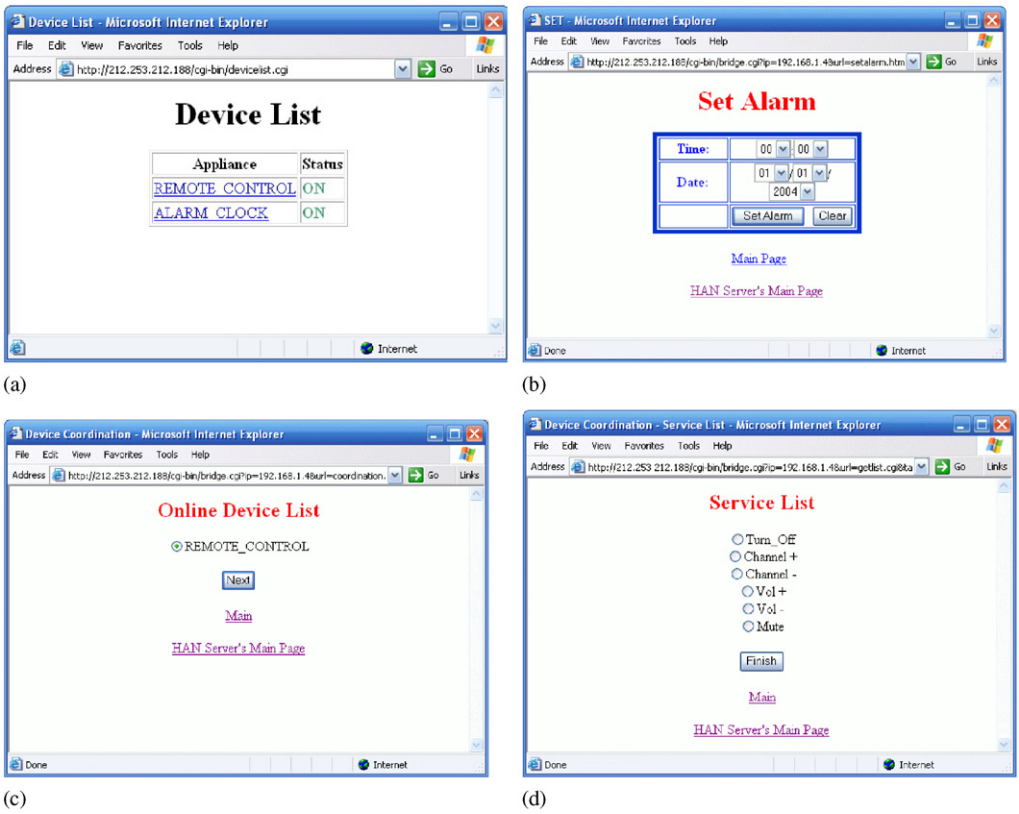


Fig. 5. Device coordination: (a) The web page showing the list of online devices is dynamically generated by the Ethernut home server. (b) The alarm clock is set via a form. (c) In device coordination online device list is get from EHS. (d) In device coordination related service information is get from the device itself.

any information on the system can easily utilize the system since all necessary configurations are done automatically.

5. Benefits and drawbacks of the HAN architecture

Embedded Web Servers approach used in this study provides several interesting benefits, which were demonstrated in the prototype system. Embedded Web Server-based HANs provide a general purpose technique, which allows users to employ a single control device with the Internet connection to interact with diverse appliances with embedded systems that he or she may encounter in his or her daily life. Moreover, web enabling appliances provides a method for interfacing to appliances without any client side programming. In addition, it provides platform independence. We based our architecture on powerful and universally available TCP, IP and HTTP protocols instead of designing special protocols for communication and interaction.

Our approach is also flexible, dynamic and adaptive. For instance, manufacturers can easily incorporate different features and functionality in their products or improve user

interfaces by designing new web pages and CGI functions. Moreover, manufacturers can perform software updates remotely. They can extend or update appliance interfaces by simply uploading new web pages to appliances' web servers. Accessibility options for helping handicapped users or multi-language features can be added to appliance interfaces. Our approach places minimal resource requirements on appliances as they must only implement a web server on the existing embedded systems. The approach allows device manufacturers to implement complex functionality, such as value-added services, even for the simplest appliances with microcontrollers.

Designing HAN compatible appliances is simple since the developers of embedded systems can use traditional programming techniques and conventions. Moreover, there is no need to implement low level components such as TCP/UDP sockets, CGI function handlers, thread manager, DHCP Client or HTTP file system. The APIs are presented completely in the prototype system. The costs related to user interface design, product documentation, training and support can be reduced considerably with our approach.

For consumers, configuration of HAN compatible appliances is completely transparent. When an appliance is plugged into the network, it obtains its network configuration automatically. Moreover, appliance interfaces as web pages are easier to interact with since they can contain labels that are more descriptive and certain multi-step tasks can be put on separate web pages where navigation between steps are accomplished by hyperlinks.

One drawback in the HAN architecture is the communication overhead related to the appliance registration. An appliance should register itself with the HAN Server in certain periods of time. In the Client side, a thread, which is dedicated especially for registration, sleeps for a certain period of time. When it wakes up, it opens a TCP socket and makes an HTTP request to the HAN Server. Therefore, there is no need to use an additional thread on the Server side. Appliance registration process makes use of existing HTTP threads in the HAN Server. Since appliance registration takes place in the local network and amount of information exchanged during registration process is about 1 KB, communication overhead becomes a less serious problem.

The current version of the HAN Server has some inherent hardware weaknesses. Firstly, the number of appliances that can be registered is not scalable enough to be used in environments that contain a very large number of appliances. Currently, the HAN Server supports 10 appliances connected concurrently. The reason for this limitation is the available RAM capacity. (The Ethernut 1.3 comes with 4 KB of Internal RAM and 32 KB of external RAM.) However, this is clearly only a problem with the current hardware implementation and, by no means, a deficiency in the architecture.

Another implementation problem is with displaying pictures. When a request to an appliance's web page comes from the Internet, a bridge function modifies the links within the web page to make these links accessible over the Internet. If the web page contains images, the bridge function cannot transfer these pictures because of the memory limitations. The Ethernut needs at least 8 KB of free RAM space to process an HTTP request. When an image is requested via the bridge function, the Ethernut enters low memory condition and all HTTP requests are dropped. The latest version of the Ethernut PCB, which comes with 512 KB of external RAM, is 2.1. If the HAN Server is imported to the latest version, the memory problems can easily be solved. The HAN Server source code is written compatible with the version 2.1.

6. Conclusions

We designed an application architecture and built and demonstrated a prototype system that can integrate web servers to everyday appliances with embedded microcontrollers to control and manage them via web pages using regular web browsers over the Internet. The system allows forming device networks such that their components can easily make use of each others' services and functions.

The unique architecture presented in this paper is used to develop the prototype system which, in turn, is used to test and demonstrate the characteristics of our design. The main contributions of this architecture are its lightweight design, automatic configuration and utilization of widely available network protocols of TCP/IP and HTTP.

Our system provides for easy control of house appliances through the Internet, which will give the freedom of "not worrying" if the oven is switched on or not when you are one button close to control it via GPRS from your cell phone. The second benefit is the coordinative structure of house devices with each other. Based on the use of well-known and stable protocols such as TCP/IP, HTTP, and PPP; the system is robust and easy to develop. It offers user-friendly, low-cost interfaces to the household devices instead of expensive and complex ones. Self-configuration features of the system make it very attractive for home environments. The implemented prototype system proves that existing devices with remote controllers can easily be integrated to the system.

This study focuses on the infrastructure for using web pages as user interfaces. We do not address web page design for better user interfaces. Therefore, the methods of generating user friendly interfaces as web pages may be investigated in the future.

References

Bettstetter C, Renner C. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol, in Proceedings EUNICE 2000.

Borriello G, Want R. Embedded computation meets the world wide web. Commun ACM 2000:59–66.

Dong-Sung K, Jae-Min L, Wook HK. In: Kwan Y, editor. Design and implementation of home network systems using UPnP middleware for networked appliances. IEEE transactions on consumer Electronics, 2002. p. 963–72.

Ethernut Hardware Manual. URL: http://www.ethernut.de/

HAVi: Home Audio/Video Interoperability URL: http://www.havi.org

Jini Community. URL: http://www.jini.org

Nakajima T. Experiences with building middleware for audio and visual networked home appliances on commodity software. ACM Multimedia, 2002.

Nichols J, Myers B. Studying the use of handhelds to control smart appliances. ICDCS'03, 2003. p. 274–79.

Nichols J, Myers B, Higgins M, Hughes J, Harris T, Rosenfeld R, et al. Generating remote control interfaces for complex appliances. Symposium on User Interface Software and Technology of the ACM, 2002. p. 161–70.

Olsen Jr DR, Jefferies S, Nielsen T, Moyes W, Fredrickson P. Cross model interaction using XWeb. In: Proceedings UIST'00: ACM SIGGRAPH symposium on user interface software and technology, San Diego, CA, 2000. p. 191–200.

Pascoe B, Salutation Architectures and the newly defined service protocols from Microsoft and Sun, Salutation Consortium, June 1999.

Ponnekanti SR, Lee B, Fox A, Hanrahan P, Winograd T. ICrafter: a service framework for ubiquitous computing environments. UBICOMP Atlanta, 2001. p. 56–75.

Rekesh J. UPnP, Jini and Salutation—a look at some popular coordination frameworks for future networked devices, California Software Labs, Technical Report, 1999.

Rettig, M. Prototyping for tiny fingers. Commun ACM, 1994. p. 21–7.
RFC 1661: The Point-to-Point Protocol.
Salutation Consortium. Salutation Architecture Specification Version 2.0c Part 1. the Salutation Consortium, June 1, 1999a.
Salutation Consortium. Salutation Architecture Specification Version 2.0c Part 2. the Salutation Consortium, June 1, 1999b.
UPnP Forum. URL: http://www.upnp.org