Design and Implementation of Cooperative Network Connectivity Proxy Using Universal Plug and Play

Raffaele Bolla[1], Maurizio Giribaldi[2], Rafiullah Khan[1], and Matteo Repetto[3]

[1] DITEN Dept. University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy
{raffaele.bolla,rafiullah.khan}@unige.it
[2] Infocom s.r.l, P.zza Alessi 2/7, 16128 Genoa, Italy
maurizio.giribaldi@infocomgenova.it
[3] CNIT, University of Genoa Research Unit, Via Opera Pia 13, 16145 Genoa, Italy
matteo.repetto@cnit.it

Abstract. Reducing the network energy waste is one of the key challenges of the Future Internet. Many Internet-based applications require preserving network connectivity for getting incoming remote service requests or confirming their availability and presence to remote peers by sending periodic keep-alive or heart-beating messages. Billions of dollars of electricity is wasted every year to keep idle or unused network hosts fully powered-up only to maintain the network connectivity. This paper describes a new approach to design and implement the cooperative Network Connectivity Proxy (*NCP*) for reducing energy waste in the ever-growing future Internet. The *NCP* is implemented using Universal Plug and Play (*UPnP*), that uses a set of protocols to allow seamless discovery and interaction between the network hosts and the *NCP*. The *NCP* allows all registered network hosts to transition into the low power sleep modes and maintains the network connectivity on their behalf. It handles basic network presence and management protocols like *ICMP*, *DHCP*, *ARP* etc on behalf of the sleeping network hosts and wakes them up only when their resources are required. Depending on the network hosts time usage model, the *NCP* can provide about 60 to 70% network energy savings.

Keywords: Green networking, energy efficiency, power proxy, power measurement, Universal Plug and Play.

1 Introduction

Green technology is one of the key challenges of the Future Internet to step into a sustainable society with reduced CO_2 footprint. The Environmental Protection Agency (*EPA*) estimated that PCs in US consumes about 2% of the overall US electricity requirement [1]. For a typical household, a single 80W PC that remains powered-up 24/7 will add about 6.5% to the utility bill [2], [3].

A. Galis and A. Gavras (Eds.): FIA 2013, LNCS 7858, pp. 323–335, 2013.

Beyond PCs, the number of other Internet connected edge devices like smartphones, tablets, IP-phones, set-top boxes, game consoles, network based multimedia equipments etc are increasing at a rapid rate. Thus, reducing the energy waste of *ICT* is becoming primarily important due to the ever increasing cost of electricity, rapid increase in the Internet connected edge devices, rapid development of the Internet-based applications, high data rates, increase in the number of services offered by the telcos and Internet Service Providers (*ISP*) and environmental concerns [4].

A recent study by Lawrence Berkeley National Laboratory (*LBNL*) has revealed that about 60% of the office computers are left powered-up 24/7 with existing power management features disabled only to maintain the network connectivity for remote access, Voice-over-IP (*VOIP*) clients, Instant Messaging (*IM*) and other Internet-based applications [5]. Normally, the Internet-based applications require preserving network connectivity for getting incoming remote service requests or receiving/responding to periodic heart-beat messages. Failing to generate/respond heart-beat messages will drop the network connection and results in the application state loss [6]. Thus, much of the energy consumed by the Internet connected edge devices is wasted as these applications don't transmit real data most of the time but just the periodic heart-beat messages [1].

This paper describes the design and implementation of cooperative Network Connectivity Proxy (*NCP*) using Universal Plug and Play (*UPnP*) protocol. The *NCP* performs key network presence and management tasks on behalf of the network hosts and allows them to sleep as long as their resources are not required [4]. The *UPnP* can be well suited approach for the design of *NCP* due to its increasing popularity and automatic seamless configuration ease that it offers. Also, *UPnP* is well suited for the residential networks and can be implemented on network based devices e.g., PCs, printers, Internet gateways, Wi-Fi access points, mobile devices etc for offering their services and communicate in a seamless way [7], [8]. Thus, the *UPnP* based *NCP* uses a set of protocols that allows the network hosts to seamlessly discover and take advantage of the services offered by the *NCP*. The implementation of *NCP* inside a residential Home Gateway (*HG*) e.g., *ADSL* switch/router that will perform the basic network based activities on behalf of sleeping network hosts is also one of the key objectives of this work.

The rest of the paper is organized as follows. Section 2 briefly presents the related work. Section 3 describes briefly the *UPnP* protocol. Section 4 describes the *NCP* concept and its possible modes of operation. Section 5 presents the design of cooperative *NCP* using *UPnP*. Section 6 describes the practical implementation of *NCP*. Section 7 presents the measurements and observations. Finally, section 8 concludes the paper.

2 Related Work

K. Christensen can be probably the first to propose the *NCP* concept for reducing the network energy waste [2]. His initial work focuses on the *NCP* design for on-board *NIC* and *LAN* scenarios. He also analyzed the network traffic in university dormitory during idle and busy periods to forecast the *NCP* importance [1]. The *NCP* design on host's *NIC*, application specific wake-up, preserving *TCP* connections and strategies to increase the host's sleep period are addressed in [5]. The concept of proxy server that manages the *TCP/IP* connections on behalf of sleeping clients is proposed in [3]. The state of art work pointing key challenges in the *NCP* design is presented in [4]. It also addresses future possibilities for extending the *NCP* functionalities and capabilities.

Some preliminary work on embedding *NCP* functionalities within the host's *NIC* is proposed in [1]. Agarwal et al. proposed a low power *USB* based architecture called Somniloquy that embeds a low power processor in the PC's network interface and runs an embedded *OS* [9]. The Somniloquy uses application specific stubs to maintain the presence of applications. The proxy designs for applications like Gnutella *P2P* file sharing and Jabber clients are proposed in [10] and [11], respectively.

S. Nedevschi et. al. in [12] have classified *NCP* into four types based on its treatment to network traffic from different protocols and applications. They also presented the energy savings achieved by different proxies in home and office environment. The concept of *Selective Connectivity* was introduced in [13] that defines several degrees of connectivity, ranges from disconnection to full connection. Further, the fast and efficient hardware based packet classification that can easily sustain on high data rate links is proposed in [6].

This paper proposes the design of cooperative *NCP* that is well suited for the *LAN* scenario. It uses *UPnP* protocol to achieve auto configuration and seamless communication between the *NCP* and its clients. Our *UPnP* based approach can also be quite useful to easily extend the *NCP* functionalities for other network devices e.g., copiers, printers, scanners etc.

3 Overview of UPnP

The *UPnP* technology is designed to support zero-configuration, invisible networking and automatic discovery of the network devices. It has the potential to be widely deployed in the residential networks and will get its full penetration in home network devices in near future [7]. The *UPnP* Device Architecture (*UDA*) classify the devices into two general categories: controlled device (*CD*) (or simply 'device') and control point (*CP*) [8]. A *CD* in a broad sense performs the role of a server and responds to the request sent by the *CP*. Both the *CP* and the *CD* can be implemented on any platform like PCs and embedded systems. Fig. 1 shows the generic *UPnP* scenario. The generic work flow consists of six steps: *(i) Addressing:* The *UPnP* device gets an *IP* address. *(ii) Discovery:* The *CP* and *CD* use Simple Service Discovery Protocol (*SSDP*) to advertise/discover their presence. *(iii) Description:* The *CP* retrieves the *CD*'s description from the *URL* provided by the *CD* in the discovery message. *(iv) Control:* The *CP* sends action to the services offered by the *CD*. *(v) Eventing:* The action performed by the *CD* may cause changes in state variables value which represent the *CD* current status. The service publishes updates to the *CP* about these changes using eventing messages. *(vi) Presentation:* The *CP* can retrieve the *CD* page from the presentation *URL* of the *CD* and load it into a browser.

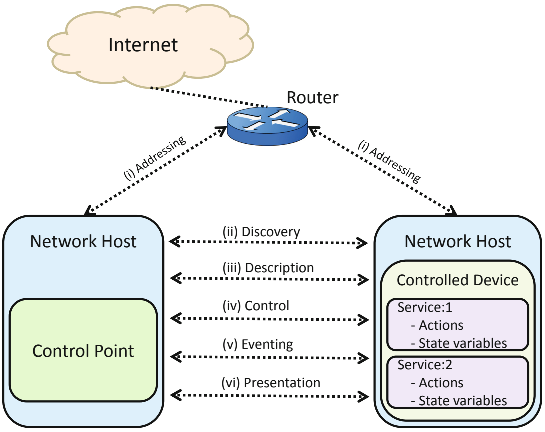

Fig. 1. Generic *UPnP* communication scenario

4 Network Connectivity Proxy

The *NCP* uses a low power entity that can maintain the network presence for high power devices and smartly make the high power devices to transition into low power sleep and active modes [4]. The *NCP* encourages the devices to enter into low power modes during idle periods that would otherwise be left powered up by the users only to maintain the network presence. The *NCP* allows automatic waking up of the devices from low power modes only when it is truly necessary [5].

4.1 Overview of NCP

The network host transfers its proxiable state information to the *NCP* before entering into sleep mode. The *NCP* starts functioning by generating/responding to the routine network traffic on behalf of sleeping host. It impersonates the sleeping host as long as a packet or new *TCP* connection request is received that requires host resources. It wakes up the sleeping host by sending Wake On LAN (*WOL*) packet (also known as magic packet) and transfers the presence state back to the host [2]. Generally, the *NCP* performs three basic tasks on behalf of sleeping host: (i) *NCP* maintains the *MAC* level reachability by generating/responding to the *ARP* requests that are intended for the sleeping network hosts. (ii) *NCP* maintains the network level reachability of sleeping host by maintaining the presence of its *IP* address in the network. It accomplishes this task by sending periodic

DHCP lease requests and responding to network presence messages e.g., *ICMP* ping requests etc. (iii) *NCP* maintains the application-level reachability for the sleeping network host. It accomplishes this task by allowing to establish new *TCP* connections and generating/responding to the network-based applications periodic heart beat messages.

4.2 NCP Types

There are two different modes of operation for the *NCP* [14].

1. *Invisible NCP:* The *NCP* doesn't advertise its presence in the network. It invisibly guesses about the power state of network hosts from the traffic analysis. Invisible *NCP* does not require any changes to the hosts and/or application servers. Since it doesn't communicate with the hosts, it cannot verify the host presence in the network while sleeping.
2. *Cooperative NCP:* Cooperative *NCP* announces its presence in the network and communicates directly with the network hosts. Thus, cooperative *NCP* requires a software on the hosts that can be used for the communication with the *NCP*. The *NCP* and host exchange two types of messages: application specific and power state messages. Application specific messages contain information about the application connections and routine heartbeat messages while power state messages include wakeup and sleep notifications.

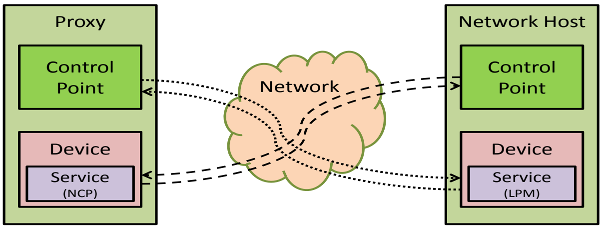

Fig. 2. *UPnP* based *NCP* generic view

5 Design of Cooperative Network Connectivity Proxy

This section describes the design of cooperative *NCP* using *UPnP*. The generic *UPnP* scenario consists of *CP* and *CD* implemented on two different network entities. The *CP* sends control messages to invoke a specific action provided by the services inside *CD*. The *CD* state variables value may change as the result of action performed. The *CD* informs all registered *CPs* about the changes in state variables value. The generic design of *UPnP* based *NCP* is shown in Fig. 2 and basic functional view is shown in Fig. 3. Both, the proxy and network host implement *CP* as well as *CD* with logical services. The proxy is implemented on the residential *HG* and has the capability to maintain the network presence for high power devices in the Home Area Network (*HAN*).

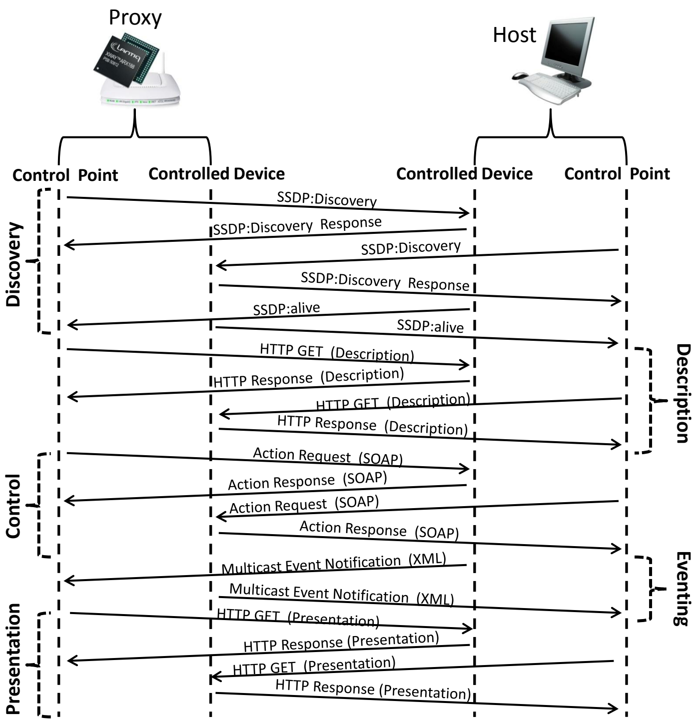

Fig. 3. *UPnP* based *NCP* functional view

Proxy's *CD* implements a network connectivity service and the network host's *CD* implements a Low Power Management (*LPM*) service. Proxy's *CP* can invoke an action defined by the power management service inside the host's *CD*. The result of these actions will change the power state of the network host. Similarly, network host's *CP* can invoke an action defined by the network connectivity service inside the proxy's *CD*. The result of these actions will define different network presence and management capabilities, that will automatically activate when the network host enters into sleep mode. These actions include:

1. *Wake-On-Connection:* Wake up the sleeping network host when a new connection attempt at the specific protocol and port is received.
2. *Wake-On-Packet:* Wake up the sleeping network host on receiving a specific packet.
3. *Send-Reply-On-Packet:* Send reply on receiving the specified packet e.g., responding to heartbeat messages.

Fig. 4 shows the flow-chart of *NCP* functionality. Since we have designed cooperative *NCP*, the proxy and the network host exchange power state information and *UPnP* control messages. Before the host enters into sleep mode, it registers the proxying request along with the required actions at the proxy. The proxy examines the power state of network host and enables packet sniffing and proxying as soon as the host enters into sleep mode. Fig. 4 also depicts the functionality of packet processing unit that performs appropriate action on each received packet addressed to the sleeping host.

6 Implementation of the NCP

The cooperative *NCP* was implemented in linux operating system on a standard PC. The *NCP* software was written in *C++* programming language using *QT* libraries. The *QT* libraries provide rich set of functions to easily perform network programming. The *UPnP CPs* and *CDs* were implemented using Herqq *UPnP* (*HUPnP*) libraries that is compliant with *UPnP* specification v1.1. The key component of *UPnP* based *NCP* is the design of services and actions. Actions work on state variables that represents the state of the *CD* at runtime. Our cooperative *NCP* requires two-way *UPnP* implementation. Thus, the network host and proxy, both contains *CP* as well as *CD*. On the proxy side, *PCAP* libraries were also used to sniff packets intended for the sleeping network hosts.

A linux kernel module was developed for the network hosts to automatically detect changes in their power state. This kernel module communicates the power state changes to the network host's *CD* that modify the corresponding state variables. The host's *CD* keeps the proxy's *CP* updated about the state variables value using *UPnP* event notification messages. The host's *CP* is used to register desired actions at the proxy's *CD*. The proxy starts proxying and creates packet sniffers based on registered actions as soon as the host enters into sleep mode.

The proxy implemented *ARP* spoofing functionality that broadcast gratuitous *ARP* reply packets to bind its *MAC* address with the sleeping host's *IP* address in the *ARP* tables of network hosts/routers. The proxy forwards the sniffed packets intended for the sleeping network hosts to the packet processing unit. The packet processing unit analyses the packet and determines the appropriate action. The proxy implemented response packets for basic network presence and management protocols like *ARP*, *ICMP* echo etc. The proxy also implemented *WOL* packet that is sent to the sleeping host when proxy sniffs a packet that requires host's resources e.g., new *TCP* connection attempt etc.

One of the objectives of this research is to implement *NCP* inside residential *HG* e.g., *ADSL* switch/router etc. Switch/Router can be the optimum place for proxy implementation and don't cause any significant increase in network energy consumption as it remains powered up 24/7. Our experiment was successfully conducted on Lantiq XWAY VRX288 (PSB 80920) v1.1 evaluation board that uses embedded linux 2.6.20. This XWAY VRX288 board embeds a 32 bit Microprocessor without Interlocked Pipeline Stages (*MIPS*) processor and has *CPU* frequency of 500 MHz, 32 KB boot *ROM*, and 16 bit DDR-2 32MB memory.

The *NCP* software was cross compiled for Lantiq evaluation board using cross compiler tool chain generated for *MIPS32* processor. The *NCP* functionalities were successfully evaluated on XWAY VRX288 board in our LAB environment.

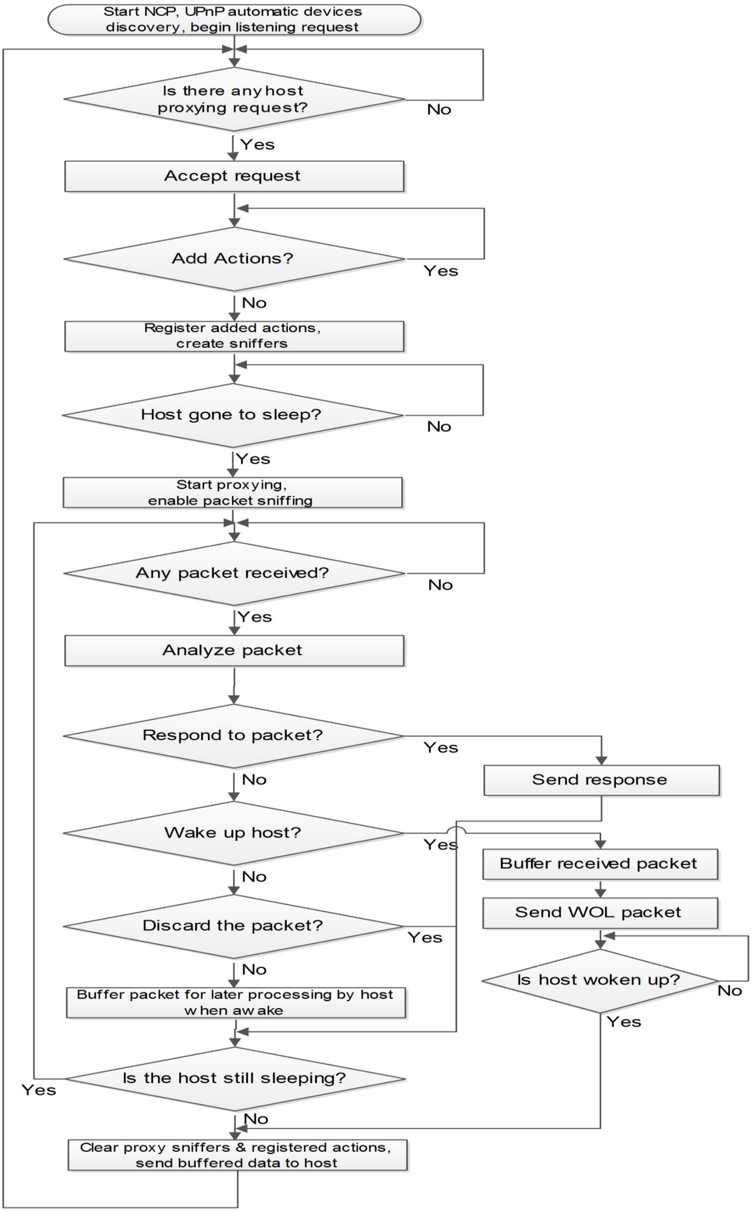

Fig. 4. *NCP* functional flow-chart

7 Measurements and Observations

Some benefits of the *NCP* and its impact on network performance and applications is presented in [12]. This paper presents the performance of *NCP* in different realistic scenarios and addresses its network overhead due to UPnP signalling. Table 1 presents the Wireshark statistical results to evaluate the network overhead. The test was performed for duration of 712 seconds considering only one *NCP*'s client that announces it presence periodically, gets registered with *NCP*, registers Wake-On-Connection action, transitions into sleep and wake-up states and finally de-registers with *NCP*. From Table 1, it can be observed that most of the packets are exchanged during steady state condition while average packet size is the smallest which are mostly the *CD*'s periodic advertisements. The avg. packet size is quite large during discovery phase as *CPs* on both sides download the *CD*'s *XML* description file.

Table 1. Traffic overhead due to *UPnP* signaling

Event Type	No. of Packets Exchanged	Total Bytes Exchanged	Avg. packet size [Bytes]
Discovery & Description	130	34678	266.75
Steady State	650	68154	104.85
Action Registration	46	5976	129.91
Power State Notification	43	4929	114.62
De-registration	80	17508	218.85

The *NCP* performance was evaluated considering two different realistic scenarios, (i) *Scenario 1: NCP* application on the *HG* covers for its sleeping client while third party host located in the same *HAN* tries to access *NCP*'s client. (ii) *Scenario 2: NCP* application on the *HG* covers for its sleeping client while third-party host tries to access *NCP*'s client from outside *HAN* (*NCP* lies between its client and third-party host). The *ICMP* echo test results averaged over 3 trials are shown in Table 2. Few packets are lost during wake-to-sleep (WTS) and sleep-to-wake (STW) transitions in scenario 1 as packets diversion through gratuitous *ARP* takes some time to update *ARP* caches of network devices. There is no loss in scenario 2 as *NCP* lies in path between its client and third-party host (No traffic diversion required). Similar considerations also hold for packet duplication. The *NCP*'s client takes some time to inform *NCP* about wake-up and stop *NCP* service, thus resulting in few *PING* packets replied by both *NCP* and its client. The Wake-On-Connection test results are also shown in Table 2 in which *NCP* client registers a wake-up request at *NCP* for Secure SHell (*SSH*) remote connection application (*TCP* port 22). The *NCP* latency is calculated as the time *NCP* takes to send *WOL* packet after receiving new *SSH* connection request for its client. The *SSH* response time is evaluated as the time when *NCP* receives and buffers the first *SSH* request packet to the time when *SSH* response is sent by the *NCP*'s client after wake-up. The *NCP* forwards the buffered *SSH* requests as soon as it's client wakes-up. While host wake-up time represents the time interval between sending *WOL* packet and first update packet sent by host after wake-up. Scenario 1 values are a bit smaller as the host receives another *SSH* request from third-party host before it receives buffered *SSH* requests from *NCP*. Whereas, *NCP* buffers continuously *SSH* requests in scenario 2 until it receives wake-up notification from its client.

Table 2. Experimental Results

	ICMP Echo Tests				Wake-On-Connection Tests		
	No. of lost PING responses		No. of duplicate PING responses		SSH response time (seconds)	Host wake-up time (seconds)	NCP latency (milliseconds)
	WTS	STW	WTS	STW			
Scenario 1	4	2	0	0	6.35	5.88	0.71
Scenario 2	0	0	1	4	10.89	10.69	0.74

Table 3. Power level measurements in ACPI S0 and S3 states

Network Host	Power Requirement (W)	
	ACPI S0 state	ACPI S3 state
PC1: Motherboard SuperMicro X8DAH+-F, two CPU Intel Xeon X5650, 6 GB RAM	146	8
PC2: Motherboard P5QPL-M, Intel CPU E5400 and 4 GB RAM	55	3
PC3: Intel Atom D510, 1 GB of RAM	22	3
Notebook1: Dell Inspiron 910	19	1.2
Notebook2: Dell Latitude D531	23.5	2
Notebook3: Toshiba Satellite A205	21.4	1.46

Table 3 represents the power measurements performed for three desktop computers and notebooks in *ACPI* S0 and S3 states. It is obvious that both desktop computers and notebooks consume much less energy in sleep state (S3:suspended to *RAM*) compared to powered-up state(S0). Table 4 shows the expected energy savings by considering average values from Table 3. It is obvious that future deployment of *NCP* in the network can provide on average 438 kWh/year energy savings for a desktop computer and 102 kWh/year energy savings for a notebook that correspond to the savings of 96.36 Euro/year and 22.5 Euro/year, respectively. The calculations were performed using 22 cent/kWh as the average cost of electricity in Europe. An organization with 10,000 PCs will be able to save about 0.96 million Euro/year for desktop computers and 0.225 million Euro/year for notebooks. Also, table 4 provides energy savings for the whole world using an estimate of 815 million in use desktops and 164 million in use notebooks during the year 2008. Table 4 does not consider increase in power consumption due to running the *NCP* service on *HG*. Table 5 presents the Lantiq XWAY VRX288 experimental *HG* power consumption in different cases by considering two hosts connected to it. It can be observed that the *NCP* service causes negligible increase in *HG's* power consumption. Thus, *NCP* has the potential to provide billions of Euro savings every year in the world.

Table 4. Estimated energy savings

Measurements/Estimates	Desktop	Notebook
'ON' Power (W)	80	22
'Sleep' Power (W)	5	2
Power savings (W)	75	20
Actual usage (hour/day)	8	10
Actual usage (hour/year)	2920	3650
Full time 'ON' (hours/year)	8760	8760
Full time 'ON' (kWh/year)	700.8	192.72
Actual use + sleep when idle(kWh/year)	262.8	90.52
Savings for 1 (kWh/year)	438	102.2
Savings for 1 (Euro/year)	96.36	22.5
Savings for 10,000 (Million Euro/yr)	0.96	0.225
Worldwide savings (Billion Euro/yr)	78.5	3.7

Table 5. Home gateway power consumption

No host connected (NCP inactive)	Hosts connected (NCP inactive)	Hosts connected (NCP active)	Hosts connected, one sleeping (NCP active)	Hosts connected, one sleeping (NCP active & managing PING)
4.78 W	5.50 W	5.55 W	5.04 W	5.07 W

8 Conclusions

The *NCP* is a useful approach that allows high power network hosts to sleep when idle while maintaining their virtual presence using a low power network entity. This paper has addressed the generic structure of *NCP*, its basic functionalities and described a new approach for the design and implementation of *NCP* using *UPnP* protocol. The *UPnP* is in fact a well suited approach to implement the *NCP* for *HAN*. Furthermore, the *UPnP* based *NCP* concept for PCs can be easily extended for the power management of other local network devices e.g., *UPnP* printers, copiers and scanners etc. To achieve maximum possible energy savings, this paper has addressed the *NCP* implementation inside *HGs* (routers/switchs).

The future work will focus on the development of *NCP* that will also embed the capabilities to preserve open *TCP* connections on behalf of sleeping network hosts. Also, due to large number of network based applications with different type of heart-beat messages, the future work will also focus on the development of application independent *NCP*. The *NCP* has potential of providing great economic savings in the ever-growing future Internet.

Acknowledgment. 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

1. Christensen, K., Gunaratne, P., Nordman, B., George, A.: The Next Frontier for Communications Networks: Power Management. Computer Communications 27(18), 1758–1770 (2004)
2. Jimeno, M., Christensen, K., Nordman, B.: A Network Connection Proxy to Enable Hosts to Sleep and Save Energy. In: IEEE International Conference on Performance, Computing and Communications Conference, IPCCC (December 2008)
3. Christensen, K., Gulledge, F.: Enabling Power Management for Network-Attached Computers. Int. Journal of Network Management 8(2), 120–130 (1998)
4. Khan, R., Bolla, R., Repetto, M., Bruschi, R., Giribaldi, M.: Smart Proxying for Reducing Network Energy Consumption. In: IEEE International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS (July 2012)
5. Gunaratne, C., Christensen, K., Nordman, B.: Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed. Int. Journal of Network Management 15(5), 297–310 (2005)
6. Sabhanatarajan, K., Gordon-Ross, A., Oden, M., Navada, M., George, A.: Smart-NICs: Power Proxying for Reduced Power Consumption in Network Edge Devices. In: IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2008 (April 2008)
7. Goland, Y., Cai, T., Leach, P., Gu, Y.: Simple Service Discovery Protocol/1.0 Operating without an Arbiter, in draft-cai-ssdp-v1-03.txt, IETF Draft (October 28, 1999)
8. UPnP forum (2012), http://www.upnp.org
9. Agarwal, Y., Hodges, S., Chandra, R., Scott, J., Bahl, P., Gupta, R.: Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage. In: 6th ACM/USENIX Symp. on Networked Systems Design and Implementation (NSDI 2009), Boston, MA, USA (April 2009)
10. Jimeno, M., Christensen, K.: A Prototype Power Management Proxy for Gnutella Peer-to-Peer File Sharing. In: Proceedings of the IEEE Conference on Local Computer Networks, Dublin, Ireland, October 15-18 (2007)
11. Werstein, P., Vossen, W.: A Low-Power Proxy to Allow Unattended Jabber Clients to Sleep. In: Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2008 (December 2008)

12. Nedevschi, S., Chandrashekar, J., Liu, J., Nordman, B., Ratnasamy, S., Taft, N.: Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI). USENIX Association, Berkeley (2009)
13. Allman, M., Christensen, K., Nordman, B., Paxson, V.: Enabling an Energy-Efficient Future Internet Through Selectively Connected End Systems. In: Sixth Workshop on Hot Topics in Networks, HotNets-VI (November 2007)
14. Nordman, B., Christensen, K.: Improving the Energy Efficiency of Ethernet-Connected Devices: A Proposal for Proxying. Ethernet Alliance White Paper (September 2007)