Saving Energy by Delegating Network Activity to Home Gateways

Bolla, R., Chiappero, M., Khan, R., & Repetto, M. (2015). Saving Energy by Delegating Network Activity to Home Gateways. IEEE Transactions on Consumer Electronics, 61(4), 445-453. DOI: 10.1109/TCE.2015.7389798

Published in:
IEEE Transactions on Consumer Electronics

Document Version:
Publisher's PDF, also known as Version of record

Saving Energy by Delegating Network Activity to Home Gateways

Raffaele Bolla, *Member*, IEEE, Marco Chiappero, Rafiullah Khan, and Matteo Repetto

Abstract — *Today, an ever-increasing number of devices has networking capability. The main implication of this fact is that such devices are often left fully powered yet idle just to maintain their network presence, hence leading to large energy waste. This ultimately results in higher electricity cost for consumers.*

This paper tackles an effective mechanism to reduce energy waste of consumer electronics, by boosting the usage of low-power states available in most devices. The main concept is to delegate background networking routines to home gateways, which are today available in most homes and offices. The paper describes the functionality and the software architecture to be implemented by home gateways and consumer electronics, reports performance evaluation on a working prototype, and provides estimation of potential benefits for consumers[1].

Index Terms — Energy saving, Home Gateway, UPnP interface, Green Home Networks.

I. Introduction

The widespread usage of consumer electronics has raised several concerns about their energy consumption: indeed, estimations say that 12% of household electricity in the US is ascribable to consumer electronics [1]. The most unsettling fact is that a large share of the energy is wasted by idle states (i.e., devices are fully powered but do not perform any useful operation). To reduce their power consumption, many appliances already provide power management features and low-power states (often referred to as *standby* or *sleeping*), where some hardware components are powered off, but the whole system remains ready to quickly resume operation (e.g., this is commonly used on TVs, DVD recorders, set-top boxes, computers).

Power management is currently rather ineffective for networked devices, because they should remain available on the network most of the time. This usually happens for several purposes: to retrieve updates, to be accessed remotely, to interact with other devices. Maintaining "network presence" implies several background recurring tasks, like answering networking protocols (for instance, address resolution, resource discovery, control and diagnosis); this demands additional functions in standby/sleep states and eventually raises the power drawn in such states by several watts, making them far less efficient than expected. There are even cases where the standby power consumption is pretty the same as normal operation, as often happens for set-top boxes [1], [2]. Running devices this way results in large energy waste, which amounts to several gigawatts worldwide [3].

Computers provide a symptomatic example [1], [4]. Many computers are left on even when nobody is attending them (e.g., at night-time and on weekends), for several reasons: to discover other devices and services and to be discoverable in turn, to refresh soft states maintained by some applications (for example, waiting for a file transfer in peer-to-peer applications), to retrieve software updates, and to be ready to answer occasional requests for services/resources (like remote connections, web requests, file retrieval). In addition, some computers in offices remain powered up to run network services like *Network File System*, *Network Address Translation*, *Firewall*, *Web Server*, *Domain Name Server*, and so on. The need to be present and available on the network leads to poor usage of low-power modes: studies have revealed that about 60% of office computers are left powered-up 24/7 with existing power management features disabled, 36% are turned off and only 4% have power management enabled [5].

Energy-efficient consumer electronics would reduce both the cost of their usage and their carbon footprint, with obvious benefits for individuals and for the society. The design of efficient networked devices entails two main issues: on the one hand, reducing the power drawn in standby/sleep [6], and, on the other hand, boosting an intensive usage of that state. The latter is a great challenge, since it affects the architecture of devices and the way they interact with the network.

The *Network Connectivity Proxy* (NCP) is an effective mechanism to exploit low-power states: it runs background networking routines on behalf of other devices and wakes them up when necessary, thus allowing them to sleep (or to remain in similar low-power states) for long periods of time. To be really effective, an NCP implementation should add negligible or minimal power consumption to the whole system. Home/access gateways (HGs) are perfect candidates to perform such function, since they usually are active most of the time to provide Internet connectivity and related services.

[1] R. Bolla is with the Department of Electrical, Electronic and Telecommunication Engineering and Naval Architecture (DITEN), University of Genoa, Italy (e-mail: raffaele.bolla@unige.it).

M. Chiappero is with the Department of Electrical, Electronic and Telecommunication Engineering and Naval Architecture (DITEN), University of Genoa, Italy (e-mail: marco@reti.dist.unige.it).

R. Khan is with the Department of Electrical, Electronic and Telecommunication Engineering and Naval Architecture (DITEN), University of Genoa, Italy (e-mail: rafiullah.khan@unige.it).

M. Repetto is with the National Inter-University Consortium for Telecommunications (CNIT), Research Unit of Genoa, Italy (e-mail: matteo.repetto@cnit.it).

Contributed Paper
Manuscript received 09/30/15
Current version published 12/29/15
Electronic version published 12/29/15.

This paper describes the architectural components that should be integrated in consumer electronics to implement the NCP function: the NCP agent, which runs on HGs, and the client agent, which is used by devices to delegate their networking routines. In addition, this paper discusses the effectiveness and the scalability of the solution, together with an assessment of potential benefits for consumers.

The paper is organized as follows. Section II introduces the concept of NCP and reviews previous activity in this field. Section III sketches the conceptual architecture to implement the NCP on the HG and describes the basic set of functions. Section IV discusses the software architecture for the client-side. Section V outlines the basic requirements for a standard control interface between the NCP and its clients. Section VI reports performance evaluation, highlighting potential benefits for consumers. Finally, Section VII gives the conclusions.

II. CONCEPT, BACKGROUND AND RELATED WORK

The main purpose of the NCP is to hide the current power state of covered devices to other hosts, so that they appear present and available, even when they are in low-power (unresponsive) modes, often referred to as "*sleep*" states. This is accomplished by running background networking routines on behalf of such devices. Examples of routines that can be delegated to the NCP include:

- responding to basic connectivity protocols, like the *Address Resolution Protocol* (ARP) [7], the *Internet Control Message Protocol* (ICMP) [8], the *Internet Group Management Protocol* (IGMP) [9], and NetBIOS;
- confirming the application's presence and status by sending and responding periodic messages, which are exchanged to check whether the remote peer is still running (such messages are usually indicated as *keep-alives* or *heartbeats*);
- waking up devices when necessary, e.g., to access data stored on disk or to run local applications.

The primary benefit from the implementation of the NCP concept is the possibility to put devices to sleep, without losing their network presence. This way, great energy saving can be achieved, by avoiding devices to remain "idle" for long periods of time. The NCP processes packets intended to sleeping devices and carries out any mechanical activity to confirm their availability on the network; further, it wakes them up when their involvement is really needed (e.g., other hosts request services like file access, video streaming). In this respect, the NCP is a more effective solution than other technologies like Wake-on-LAN, which only wakes hosts up on request.

However, running an NCP brings two main concerns. First, transitions between power states imply some latency, due to operations to suspend and to resume the hardware. Second, the device hosting the NCP service consumes power, hence there is the risk that the energy to run the NCP could be larger than that saved by putting devices to sleep.

The NCP can be implemented as a software application. Running this software on the general-purpose CPU present on HGs for management purposes is a very effective solution, since such equipment must always remain powered on to provide network connectivity, thus the additional power to run the NCP is negligible [10]. However, this kind of devices has low computing capabilities, which may turn into poor performance when covering for a large number of devices.

Most implementations of the NCP concept have focused on the *Transmission Control Protocol* (TCP) [11] and management protocols like ARP, ICMP, IGMP [3], [12], [13]. There have also been a few proposals for proxying high-level protocols and specific applications, like UPnP, Gnutella, Jabber [14]-[16]; some authors have even proposed to run simplified versions of applications [17] or their virtual images instead of specific routines [18].

Apart the HG, other deployment alternatives have been considered for the NCP service, including smart Network Interface Cards (NICs), network equipment (like switches and access points), and standalone devices. Each of these solutions leads to different considerations about power consumption and processing capabilities [19].

Proxying network connections on the HG was already proposed by Park *et al* [20], although their implementation focused on the integration with power-saving mechanisms at the HG and did not provide any detail about the NCP architecture and the real benefits for consumers. In this paper, the preliminary architecture depicted by Bolla *et al* [10] is extended, and a thorough performance analysis is conducted to assess the effectiveness of running the NCP on HGs.

III. ARCHITECTURE FOR THE HOME GATEWAY

The *Home (Access) Gateway* (HG), also known as *Residential Gateway*, is a telecommunication device used to connect local area networks in households and small offices to a wide area network (typically the Internet), by a broadband connection, e.g., *Digital Subscriber Line* (DSL), cable, mobile network. HGs usually have several broadband interfaces and provide Quality of Service (QoS) features, in order to simultaneously support different types of services (for instance: data, voice, gaming), though they are less expensive and have lower performance of *Customer Premise Equipment* (CPE) used by large enterprises.

The typical architecture for HGs includes several network interfaces, hardware accelerators and a general-purpose low-power processor. Hardware accelerators switch or route packets, manage packet priority, and provide security features. The general-purpose processor runs an embedded *Operating System* (OS), which executes control applications and services. Hardware accelerators directly handle most of the network packets exchanged between the local network and the Internet; they only deliver to the OS packets intended to the HG and packets they are not able to process. This structure allows both the performance needed to handle high volumes of traffic and the flexibility to run custom applications and services.

Fig. 1 shows how the NCP function can be integrated in a typical HG architecture. The framework is made of the following logical elements: a database of *rules*, *packet*

processing, *scheduling*, *packet filtering*, the *control interface* and *raw sockets*. Besides that, there is the main logic that supervises and organizes all the components (*Orchestrator*). All components can be implemented in user space, building a modular application that can run on commodity embedded Operating Systems.

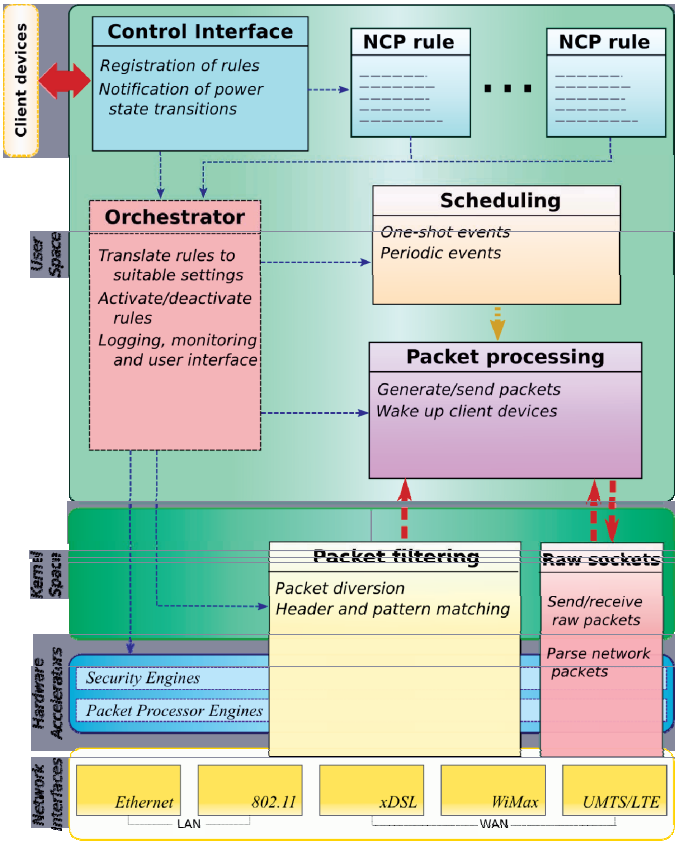


Fig. 1. NCP architecture for home gateways.

NCP rules contain the description of the behavior requested by clients. They consist of two parts: the *condition* and the *action*; the former specifies the event that triggers the execution of the latter. *Conditions* either select packets addressed to client devices or define time schedules. In the first case, they are given as matching criteria on packets' content; in the second case, they are time intervals. *Actions* are the routines delegated to the NCP; they include processing, buffering, generation and sending of network packets, and waking up sleeping devices. A behavioral rule is "active" when the corresponding client is sleeping.

Packet filtering inspects packets and catches those that match the conditions in active rules. The inspection process considers header information (source and destination addresses, protocol, source and destination ports, protocol-specific flags and options) as well as packet content (bit patterns, application-specific headers and data). HGs are a forced transit point for all traffic exchanged with external hosts, though they might not see packets exchanged between local devices. Packet diversion towards the NCP is therefore necessary for transparent operation.

Scheduling triggers *actions* as the time elapses. Triggering could be one-shot (i.e., just one trigger occurs) or periodic (i.e., many triggers are generated).

Packet processing performs the *action* of each NCP rule (i.e., it runs a given software routine), when triggered by the relative *condition*. The NCP should account at least for the following kinds of operations: waking devices up, parsing and/or modifying incoming packets, building new packets.

Raw socket is the API that enables to read and to write network packets directly at the lowest layer, thus bypassing the entire built-in networking stack. Raw sockets are mostly needed to build network packets on behalf of client devices (i.e., with a different IP address); this operation would otherwise be prohibited by the OS.

The Control Interface receives commands from clients, namely registration of their rules and notification of their power state transitions; it is discussed in Section V.

Finally, the Orchestrator is responsible for updating the database of behavioral rules and for activating/deactivating them every time a device enters/exits sleep mode, respectively. Such operations are triggered by the control interface, in order to be synchronized with the clients' power status and to cover for them in a seamless way. The Orchestrator also instructs hardware accelerators to deliver to the kernel (and, consequently, to the filtering engine) all packets intended to client devices that the NCP is currently covering for.

The framework must also account for additional peculiarities that derive from the HG role. For example, being at the boundary between local sites and the Internet, HGs usually translate between private IP addresses and the public addresses of their broadband interfaces. Packet processing and filtering must account for the presence of *Network Address Translation* (NAT) on the HG and must behave accordingly.

The NCP function should include at least the following set of general rules:

- *ARP rule*. The NCP answers ARP queries on behalf of sleeping devices. It provides its own MAC address in such responses, in order to get all traffic addressed to the covered devices. Further, it also sends Gratuitous ARP packets when it starts and when it stops covering each device, in order to update the caches of the other hosts on the network. This rule implements the "traffic diversion" feature needed to intercept traffic intended for sleeping clients.
- *Ping rule*. The NCP answers ICMP echo-request messages on behalf of sleeping hosts.
- *DHCP rule*. The NCP periodically renews IP addresses released by a *Dynamic Host Configuration Protocol* (DHCP) server [21].
- *Wake-on-Connection (WoC) rule*. The NCP wakes sleeping hosts up when packets addressed to a given transport port are seen. Packets triggering the WoC rule must be buffered at the NCP and must be sent out once their target device has completely resumed, otherwise they would be lost.

- *TCP-KeepAlive (TCP-KA) rule*. The NCP maintains a TCP session active, by implementing the keep-alive mechanism envisioned by this protocol [11].
- *Heartbeating (HrtBt) rule*. The NCP generates both solicited and unsolicited heartbeats: solicited heartbeating is triggered by incoming messages (in a similar way to the acknowledgements sent in response to TCP keep-alive messages), whereas unsolicited heartbeats are sent periodically at predefined time intervals.

In addition to the above rules, home gateways could also feature additional operations tailored to specific protocols and applications. It is rather unlikely that HGs could include NCP extensions for any application; instead, they should allow client devices to load their own background software routines.

One of the hurdles to make remote peers unaware of the delegation process is the management of TCP sessions. In fact, breaking a TCP connection might bring side effects on the application's behavior (for example, in P2P file sharing, the client may lose its position in the waiting queue). Transparent and seamless TCP migration among different hosts requires the ability to 'freeze' on-going sessions and to 'resume' them later. This process needs support from the Operating System [22].

IV. Architecture of the NCP Client

Client devices must be able to interact with and to control the NCP: they need to register background routines with the NCP and to notify it when to start and to stop covering for them. These operations entail a couple of functions: knowledge about what routines are needed by the system and the running applications, and detection of the transitions to/from the low-power state. Transitions may happen automatically (when no activity is detected or no time-critical tasks are running on the host) or may be triggered manually by the user.

The most effective solution to tackle the above functions is a software framework that hides any detail about the service interface and the OS's power management subsystem. This approach abstracts away the communication protocol and the OS's internals and reduces the need for coding, hence acting like a sort of middleware for applications.

The architectural framework for client devices is built of two main parts (see Fig. 2): a core process and a tiny library for applications. The core is a common and system-wide process, which deals with the communication protocol(s) to delegate network activity, and which interacts with the power management subsystem. It also takes care of handling system-wide tasks at the link/network layers. The library is just a small layer that easily integrates with applications and provides a transparent gate towards the core block in terms of a streamlined and intuitive *Application Programming Interface* (API) to control the NCP.

The architecture depicted in Fig. 2 removes all the complexity away from the applications and confines any future development (mostly) in a single place, which is the core. That leads to two benefits:

- power management events are handled by one entity only, which can orderly control the pre-suspend and post-resume procedures, with a single coherent policy;
- since the application library is very simple and thin, multiple implementations for different programming languages are provided easily.

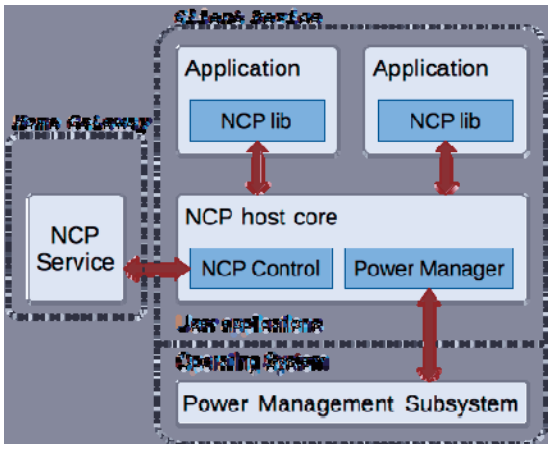


Fig. 2. Software architecture for NCP clients (right side) and interaction with the NCP (left side).

V. Control Interface for the NCP

The definition of a control interface is necessary to ensure interoperability among devices from different vendors. Such definition mainly concerns the selection of the communication paradigm and the protocol that best fit the NCP framework.

There are several protocols that can be used to control networked devices (e.g., printers and scanners, multimedia boxes, network-attached storage servers). Among them, the Universal Plug and Play (UPnP) is a versatile solution that provides zero-configuration, transparent networking and automatic discovery. UPnP allows devices to announce their presence in the network, to discover other devices, to get descriptions of and to control remote services, to subscribe for event notification, and to get high-level human-readable presentations of devices and services.

The UPnP framework and operation are described by the UPnP Architecture [23]. Basically, it distinguishes between Controlled Devices (CDs), which act as servers, and Control Points (CPs), which request services. The interaction between CDs and CPs includes:

- *Control*: CPs invoke actions provided by specific services;
- *Eventing*: CPs subscribe state variables and are notified of any change;
- *Presentation*: CDs publish available services, including the list of actions and state variables.

The NCP interface can be formalized as a UPnP service provided by the Internet Gateway device [24]. The actions of this service shall basically match the operations that client devices can demand to the NCP, plus some additional issues, like withdraw of previously registered rules, transfer of protocol or application data (e.g., TCP state information to migrate the session), and notification of sleep/active mode [25].

Notification of power state transition may also happen through other mechanisms than a specific action provided by the NCP service. For example, the UPnP Low Power architecture [26] describes functional requirements for exposing power capabilities, including the current power state. CPs can subscribe the power state variable of other devices, in order to get notified about their power state transitions. The NCP might act as CP of UPnP-LP clients to know their power status, without the need to invoke the specific NCP action; the corresponding logical architecture is shown in Fig. 3.

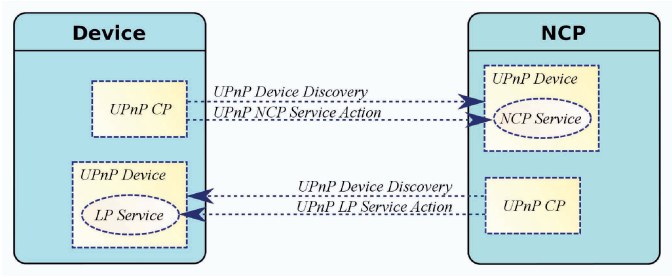


Fig. 3. Integration of the NCP with the UPnP-LP architecture. The NCP acts as CD for the NCP service and as CP for the Low Power service; clients act as CPs for the NCP service and as CDs for the Low Power service.

VI. Evaluation and Performance Analysis

A prototype of the devised NCP framework, compliant with the specification described herein, has been developed for testing; it includes the software for both the HG and client devices.

The HG platform is a commercial development board running the Linux-based *openwrt* distribution. The NCP is a software application in user-space. It provides all the basic actions listed in Section III, supports TCP migration and NAT, and is controlled by client devices through the UPnP interface described in Section V. This is an enhanced and stable release of a preliminary version of the same software [27], tailored to a commercial HG platform. The application is entirely written in C++ and exploits system calls to control low-level functions (e.g., raw sockets). It uses the *pcap* library to filter packets in an effective and portable way.

The client software detects the NCP function automatically, thanks to the discovery features of UPnP, it registers a given set of rules and notifies every power state transition. For evaluation purposes, a simple chat application was also developed, including a heartbeating mechanism based on TCP communication.

The whole software framework described above was used to evaluate the NCP concept and the effectiveness of its implementation. The main purposes were to understand whether HGs are suitable platforms to run the NCP service, and to assess the potential benefits of integrating NCP client functionality into consumer electronics. Specific issues under consideration were functional verification, performance analysis and estimation of expected energy saving and cost reduction for consumers.

A. Functional evaluation

Functional evaluation showed that all the software behaves correctly, both on the HG and on client devices. The HG was able to cover for sleeping devices in a seamless and transparent manner, thus demonstrating the feasibility of the NCP concept in a real environment. In particular, the evaluation pointed out that the NCP framework is compliant with legacy networking protocols and current Operating Systems.

The software for client devices is plain enough to be easily implemented in different languages. The integration with the power management subsystem is quite straightforward on common OSes. The most complex part is indeed the communication protocol; however, UPnP is widely available in consumer electronics. All these facts indicate that the NCP client functionality can be integrated in software and firmware of relevant consumer electronics.

B. Performance analysis

Performance analysis was conducted to assess whether typical HG hardware platforms were able to carry out NCP operations in a satisfactory way. This means client devices and other hosts on the network should be minimally affected by NCP operations. Main hardware constraints in this context are processing power and memory: HGs are usually equipped with low-end processors and small amount of volatile memory, because they just have few software tasks to carry out, but they must draw little power and must be cheap.

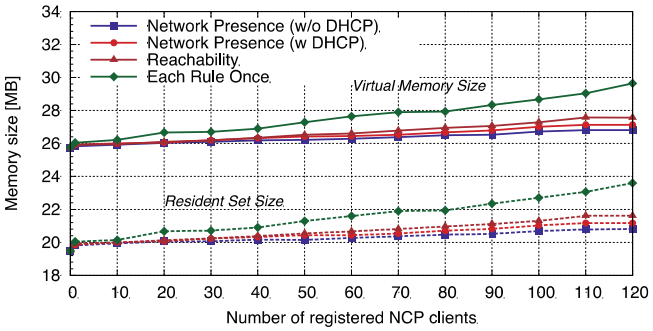


Fig. 4. Memory used by the NCP to cover for an increasing number of client devices. The graph shows both the virtual memory size (solid lines) and the resident set size (dashed lines).

The first interesting factor is the memory taken to cover an increasing number of devices. Four different sets of rules requested by client devices were considered: basic Network Presence (1 PING rule) for dynamically or statically configured hosts (namely with/without 1 DHCP rule, respectively), Reachability (1 PING, 1 DHCP and 1 WoC rules) and Each-Rule-Once (1 PING, 1 DHCP, 1 WoC, 1 TCP-KA and 1 HrtBt rules). Fig. 4 shows both the virtual address space used by the NCP software (*Virtual Memory Size*) and the actual size filled in the RAM (*Resident Set Size*). According to the measures, there is no problem to run the service on commercial platforms equipped with only 64 MB of memory.

The second constraint is CPU. Evaluations pointed out that processing is mostly required to set up the filtering engine every time a device changes its power status, and to handle packets intended to sleeping devices.

Fig. 5 shows the latency to configure the filtering engine for an increasing number of covered clients; 4 rules were registered for every device (1 Ping, 1 WoC, 1 TCP-KA and 1 HrtBt). The latency quickly rises up to several seconds, due to the behavior of the *pcap* library. In fact, this library uses Berkeley Packet Filters (BPFs), which are compiled starting from human-readable strings, in order to optimize packet inspection; however, this operation takes longer time as the string length increases (which happens with more devices and more rules). Indeed, the *pcap* library has not been conceived for frequent and dynamic changes of the filter string. When the latency exceeds some hundred milliseconds, other events could happen during the activation (e.g., other devices wake up or go to sleep, packets intended to sleeping devices are received), potentially leading to wrong or undefined behaviors. Fig. 5 suggests that HG platforms would not be able to cover for more than 20-30 devices simultaneously.

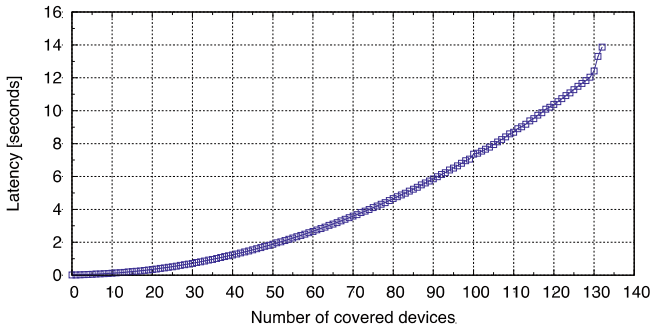


Fig. 5. Latency in setting up the filtering engine for an increasing number of covered devices.

The workload to process packets can be evaluated by 'flooding' the HG with a large amount of traffic intended to sleeping devices. Fig. 6 compares CPU usage by the NCP and by the standard networking stack, for an increasing number of ICMP echo-request packets; these packets trigger the Ping rule when the NCP is running, otherwise they are discarded. The CPU is saturated for about 3000 packets, but this amount is far beyond the traffic expected in realistic scenarios. Hence, typical CPUs mounted by HG platforms should be enough to run the NCP service without problems.

Transparent operation implies negligible effects on the traffic exchanged by the NCP with remote hosts; indeed, when the CPU usage grows, packet losses occur and the latency increases, as shown in Fig. 7, for an increasing load of ICMP echo-request packets. Losses are due to the size of the *pcap* buffer used to capture packets: it should be kept small to reduce the amount of memory allocated, but, in this situation, the buffer could get filled in before the queued packets are processed. By selecting the proper buffer size, the HG is able to respond to a large number of ICMP packets with minimal delay and negligible losses. Performance degrades in any scenario when the load exceeds 3000 packets, which corresponds to full CPU usage (as previously shown in Fig. 6).

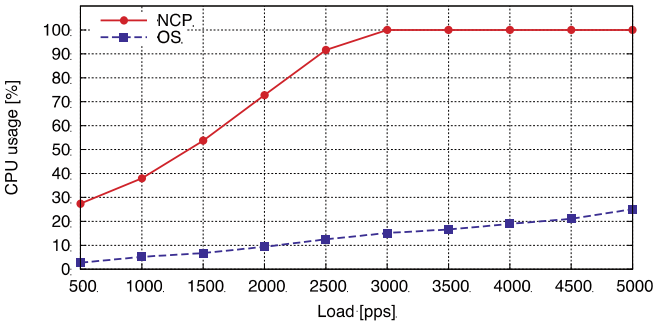


Fig. 6. CPU usage versus increasing traffic load. CPU usage is reported as percentage of maximum processing capability, traffic load is measured in packets per second (pps). The graph compares CPU usage when the NCP is running and processing packets (solid red line) and when the same packets are processed by the Operating System (dashed blue line).

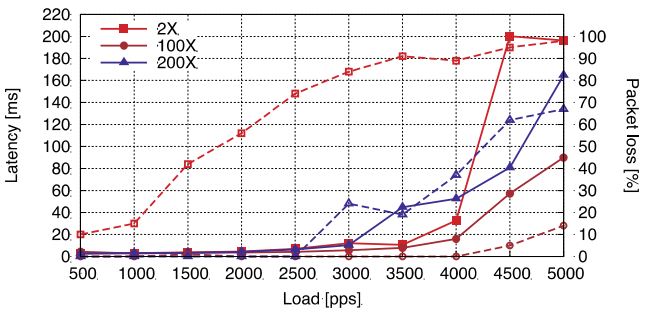


Fig. 7. Latency (solid lines) and packet loss (dashed lines) with high ICMP traffic load. The curves were obtained with *pcap* buffer sizes of 2, 100 and 200 times the basic unit of 65535 bytes.

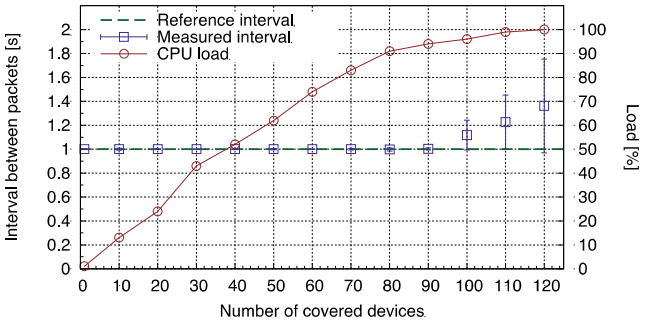


Fig. 8. Measured interval between heartbeat packets (scheduled one per second) and CPU load. The "Reference interval" shows the desired generation period (i.e., 1 second).

Finally, the tradeoff between latency and CPU usage was also analyzed for tasks that are not triggered by incoming packets, but that are scheduled periodically. Fig. 8 shows how the generation of unsolicited heartbeats affects the CPU load and the accuracy of packet timing. In this case, 10 HrtBt rules were registered for each device, leading to a very high load, far more than the traffic expected in real scenarios. Computation resources are enough to generate 900 heartbeat messages in parallel with high accuracy, which in the specific scenario correspond to 90 covered devices. Also in this case, performance degrades when the CPU comes close to saturation.

C. Expected benefits for consumers

Despite the proof of concept, it is worth evaluating the benefits for the consumer brought by the implementation of the NCP function. This service has been conceived to avoid running idle devices at full power just to maintain their presence on the network; hence, the benefits can be estimated in terms of energy saving (and implicitly cost saving) with respect to legacy operation. The evaluation should consider the following aspects:

i) usage patterns of devices, in particular, how long they usually remain idle;
ii) the set of behavioral rules available at the NCP, i.e., the type of traffic it can deal with and the ability to cover for different applications;
iii) the number of devices that can be covered by the NCP simultaneously, which is limited by the pretty low processing power and memory available in HGs.

Estimating the number of supported clients is quite straightforward: the performance analysis in Section VI.B has revealed that a typical HG is able to cover simultaneously a few dozen devices.

However, estimating the time that such devices could sleep is far more complicated, as it strongly depends on the device type, its typical usage, and the user's behavior. Data and analyses of this type are only available for computers [3], [12]; they can be used to provide a realistic assessment of the potential for energy saving in specific scenarios. TABLE I reports data about the time spent by computers in different power states, taken by past studies in home and office environments [1], [4]. A thorough investigation should also have considered other kinds of consumer electronics (printers and scanners, game consoles, media stations, etc.); however, the lack of analyses about their usage patterns would have led to arbitrary and perhaps unrealistic hypotheses, resulting in an estimation of little significance.

The first step towards the evaluation of potential energy saving consists in computing the average consumption under typical usage patterns. The total energy E_T consumed in the reference period T by a single device is given by:

$$E_T = P_A T_A + P_I T_I + P_S T_S + P_O T_O \quad (1)$$

TABLE I

POWER PROFILES FOR COMPUTERS IN HOME AND OFFICE ENVIRONEMENTS

	Active		Idle		Sleep		Off	
	[W]	[%]	[W]	[%]	[W]	[%]	[W]	[%]
Home	70	14.3	56	17.8	3.4	23.8	1.6	44.1
Office	84	4.2	64	66.9	5.3	3.3	2.5	25.6

For each power state, the first column (W) is the average power consumption and the second column (%) is the average percentage of time spent in that state.

where P_i and T_i are the power and the time spent in the i state (i=Active, Idle, Sleep, Off), respectively. Typical values for these parameters are given in TABLE I.

The NCP allows reducing the idle time and increasing the sleeping time, thus cutting down the total energy. The power budget for the NCP scenario is similar to (1), but it must also account for the additional power consumption of the HG (E_{HG}); however, such factor is shared by the number of covered devices (N). The percentage of energy saving ($\%E_{sav}$) for each device can be easily derived as:

$$\%E_{sav} = \frac{E_{sav}}{E_T} \times 100 = \frac{\alpha(P_I - P_S)T_I - E_{HG}/N}{P_A T_A + P_I T_I + P_S T_S + P_O T_O} \times 100 \quad (2)$$

where α is the percentage of reduction of the idle time due to NCP operation. The parameter α depends upon the user behavior and the NCP capability; for the kinds of rules discussed in this paper, it was estimated to be around 87% and 97% for typical office and home scenarios, respectively, under loose conditions on the transition times between the active and sleep states [12].

Fig. 9 shows the percent energy saving per device for an increasing number of clients; the range is chosen to avoid more than 1-second delay for activation (see Fig. 5). The analysis considered the two usage patterns shown in TABLE I, namely home and office, and two different hypotheses on the HG: without and with power saving capability. In the first case, the entire power drawn by the HG board in the experimental setup (13W) was used in (2), whereas, in the second case, just the standby power was considered in that formula (4.6W, according to recent research work on this matter [20]). For simplicity, E_{HG} was computed by considering the active/standby power for the entire reference period T; this corresponds to a worst-case analysis, because the contribution of the HG consumption in (2) should only account for the *additional* energy consumption, which is the energy wasted to keep the HG active just to run the NCP service, when no traffic is exchanged between the local network and the Internet.

The percentage of energy saving per device quickly rises as the number of clients increases, approaching the theoretical limits given by the usage patterns in the two scenarios. The limits correspond to the ideal case of removing all idle periods (i.e., α=100%).

In general, a larger improvement in efficiency is achieved in the office scenario, due to the presence of high-end servers, the need to run network services and less care by employees. Nevertheless, the gain is also considerable for homes, especially when energy-aware HGs are taken into account. There is just one case where running the NCP service brings a negative budget, i.e. a HG without power saving capability that covers a single device at home.

Fig. 9 indicates that about 20-35% overall energy reduction for computers is possible in homes, where the NCP is expected to cover for fewer devices, and a target of about 65-70% is realistic in offices, where there are bigger opportunities. In economic terms, considering the current average US cost of electricity for the residential and commercial sectors, the NCP allows saving about 10 $/year

per device in homes and almost 40 $/year in offices. By considering the full range of consumer electronics that could be covered by the NCP (laptops, tablets, set-top boxes, printers and faxes), their typical density per user, and the constraints on the maximum number of clients, a few dozen dollars could be saved yearly at homes and several hundred dollars could be saved for offices.

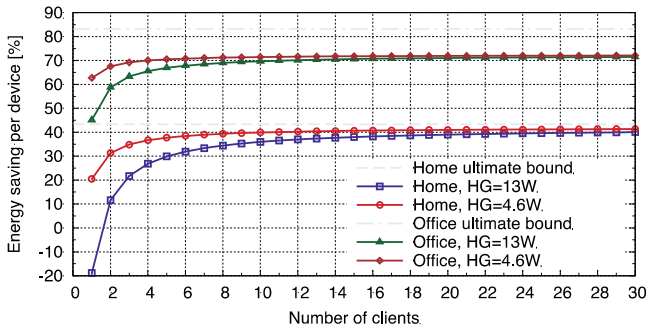


Fig. 9. Energy saving per device for different scenarios (Home/Office) and two hypotheses on the HG (with or without power saving features). The ultimate bounds are the potential saving by completely cancelling the idle periods.

VII. CONCLUSIONS

The NCP is an energy-saving framework that boosts the usage of low-power states already available in most consumer electronics; its underpinning concept is the possibility of performing background networking routines on behalf of *sleeping* devices. This paper has discussed the basic architecture, relying on HGs to cover for other devices, and has shown its feasibility and effectiveness by a working prototype and its performance evaluation.

The adoption of the NCP control interface by consumer electronics enables them to sleep when idle, without affecting their network presence. NCP clients only have simple operations to carry out (mainly registration of behavioral routines and notification of power state transitions), and the communication can exploit widespread protocols, like UPnP; hence, the implementation of the NCP framework in consumer electronics is not really a problem.

Performance evaluation has shown that a single HG can cover a few dozen devices concurrently, despite its limited processing capability. This saves substantial amount of energy and eventually results in lower operational costs for both residential and commercial users.

REFERENCES

[1] K. Roth, B. Urban, V. Shmakova, and B. Lim, "Residential Consumer Electronics Energy Consumption in 2013," *in 2014 ACEEE Summer Study on Energy Efficiency in Buildings*, Pacific Grove, USA, pp. 9-308–9-320, Aug. 2014.

[2] A. de Almeida, P. Fonseca, B. Schlomann, N. Feilberg, and C. Ferreira, "Residential monitoring to decrease energy use and carbon emissions in Europe," *in Proc. of the 4th International Conference on Energy Efficiency in Domestic Appliances & Lighting*, vol. 2, London, United Kingdom, pp. 529-544, Jun. 2006.

[3] K. Christensen, P. Gunaratne, B. Nordman, and A. George, "The next frontier for communications networks: power management," *Computer Communications*, vol. 27, no. 18, pp. 1758-1770, Dec. 2004.

[4] B. Nordman and C. Christensen, "Greener PCs for the enterprise," *IT Professional*, vol. 11, no. 4, pages 28-37, Jul.-Aug. 2009.

[5] K. Kawamoto, J. Koomey, B. Nordman, R. Brown, M. Piette, M. Ting, and A. Meier, "Electricity used by office equipment and network equipment in the US: detailed report and appendices," Tech. Rep. LBNL-45917, Lawrence Berkeley National Lab., Feb. 2001.

[6] J. Heo, C. S. Hong, and S. S. Jeon, "Design and implementation of control mechanism for standby power reduction," *IEEE Trans. Consumer Electron.*, vol. 54, no. 1, pp. 179-185, Feb. 2008.

[7] David C. Plummer, "An Ethernet Address Resolution Protocol," IETF Request For Comments 826, Nov. 1982.

[8] J. Postel, "Internet Control Message Protocol," IETF Request for Comments 792, Sep. 1981.

[9] W. Fenner, "Internet Group Management Protocol, Version 2," IETF Request for Comments 2236, Nov. 1997.

[10] R. Bolla, M. Giribaldi, R. Khan and M. Repetto, "Design of Home Energy Gateway Boosting the Development of Smart Grid Applications at Home," *in 4th International Conference on Energy Aware Computing Systems & Applications*, Istanbul, Turkey, pp. 103-108, Dec. 2013.

[11] "Transmission Control Protocol," IETF Request for Comments 793, Sep. 1981.

[12] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taft, "Skilled in the art of being idle: reducing energy waste in networked systems," *in Proc. of the 6th USENIX symposium on Networked systems design and implementation*, Boston, USA, pp. 381-394, Apr. 2009.

[13] K. Christensen and F. Gulledge, "Enabling power management for network-attached computers," *International Journal of Network Management*, vol. 8, no. 2, pp. 120-130, Mar.-Apr. 1998.

[14] J. Klamra, M. Olsson, K. Christensen, and B. Nordman, "Design and implementation of a power management proxy for Universal Plug and Play," *in Proc. of the Swedish National Computer Networking Workshop*, Halmstad, Sweden, Nov. 2005.

[15] M. Jimeno and K. Christensen, "A prototype power management proxy for Gnutella peer-to-peer file sharing," *in Proc. of the IEEE Conference on Local Computer Networks*, Dublin, Ireland, pp. 210-212, Oct. 2007.

[16] P. Werstein and W. Vossen, "A low-power proxy to allow unattended Jabber clients to sleep," *in Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dunedin, New Zealand, pp. 390-395, Dec. 2008.

[17] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, V. Bahl, and R. Gupta, "Somniloquy: augmenting network interfaces to reduce PC energy usage," *in Proc. of the 6th USENIX symposium on Networked systems design and implementation*, Boston, USA, pp. 365-380, Apr 2009.

[18] Y. Agarwal, S. Savage and R. Gupta, "Sleep Server: a software-only approach for reducing the energy consumption of PCs within enterprise environments," *in Proceedings of the USENIX annual technical conference*, Boston, USA, Jun. 2010.

[19] R. Khan, R. Bolla, M. Repetto, R. Bruschi, and M. Giribaldi, "Smart proxying for reducing network energy consumption," *in Proc. of the 2012 Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Genoa, Italy, Jul. 2012.

[20] W.-K. Park, C. Choi, I. Lee, and J. Jang, "Energy efficient multi-function Home Gateway in always-on home environment," *IEEE Trans. Consumer Electron.*, vol. 56, no. 1, pp. 106-111, Feb. 2010.

[21] R. Droms, "Dynamic Host Configuration Protocol," IETF Request for Comments 2131, Mar. 1997.

[22] R. Bolla, M. Chiappero, R. Rapuzzi and M. Repetto, "Seamless and transparent migration for TCP sessions," in *IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, Washington, DC – USA, Sep. 2014.

[23] UPnP Device Architecture, Version 1.0, Oct. 2008.

[24] UPnP Internet Gateway Device (IGD) Version 2.0, December 10, 2010.

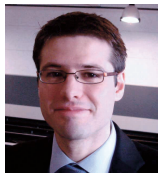
[25] R. Bolla, M. Giribaldi, R. Khan, and M. Repetto, "Design and implementation of cooperative network connectivity proxy using Universal Plug and Play," in A. Galis, A. Gavras, Eds., "*The Future Internet – Future Internet Assembly 2013: Validated Results and New Horizons*", Springer Open, LNCS, Vol. 7858, pp. 323-335, May 2013.

[26] UPnP Low Power Architecture, Version 1.0, Aug. 2007.

[27] R. Bolla, M. Giribaldi, R. Khan and M. Repetto, "Network connectivity proxy: an optimal strategy for reducing energy waste in network edge devices," *in The 24th Tyrrhenian International Workshop on Digital Communications*, Genoa, Italy, Sep. 2013.

BIOGRAPHIES

R. Bolla (S'89-M'95) received the Ph.D. degree in Telecommunications in 1994 from the University of Genoa. He is currently Full Professor at DITEN in the same University, where he is leading the Telecommunication Networks and Telematics (TNT) laboratory. He is in charge of all the International relationships of the Polytechnic School of the University of Genoa.
He has been Principal Investigator of many important research projects and contracts with both public institutions and private companies. The most recent and relevant activities include the coordination of EU FP7 ECONET project (low Energy COnsumption NETworks), the participation in the European Network of Excellence (NoE) Intermedia (Interactive Media with Personal Networked Devices), the national coordination of the PRIN project SORPASSO (flexible Software Router PlAtform for Secure Service-specific Overlay networks). He is also involved in many standardization activities in ITU-T, ETSI and IEEE (he is currently the rapporteur of a Working Item of the ETSI EE-TC).
He has co-authored over 200 scientific publications in international journals and international conference proceedings.
His current research interests are in: i) networking architectures, protocols and techniques for the smart energy grid; ii) mechanisms and techniques for energy consumption reduction in IP networks, iii) advance platform for Future Internet nodes (Flexible Software Router) iv) Software Defined Network and Network Function Virtualization Based architectures, v) advance management of user mobility in packet networks.



M. Chiappero received the 'laurea specialistica' (master degree) in Computer Science Engineering in 2012 from the University of Genoa. He was a Research Fellow at University of Genoa, covering different networking topics. He is currently working in Intel, Shannon, Ireland. He was involved in the ECONET project, funded by the European Commission under the 7th framework program.
His current research interests include computer science and energy-efficient networking.

R. Khan received his B.Sc. degree in Electrical Engineering from University of Engineering and Technology Peshawar, Pakistan, his master degree in Satellite Navigation and Related Applications from Politecnico Di Torino, Italy, and his Ph.D. degree jointly from University of Genoa, Italy and Polytechnic University of Catalonia, Spain, in 2009, 2010 and 2014, respectively. He completed his Ph.D. under a joint degree Erasmus Mundus program, funded by the European Commission. During the Ph.D., his main research area was green networking and he was involved in the ECONET project, funded by the European Commission under the 7th Framework Program.
He is currently a Postdoctoral Research Fellow with the Institute of Electronics, Communications and Information Technology, at Queen's University Belfast, in the United Kingdom. He has co-authored several papers in international journals and conference proceedings. His research interests include Ad Hoc Networking, Self-Organizing Networks and Green Networking.

M. Repetto received the Ph.D. degree in Electronics and Informatics in 2004 from the University of Genoa. From 2004 to 2009 he was a postdoc at University of Genoa. Since 2010, he is a Research Associate at CNIT.
He has been teaching many courses in telecommunication networks and network security. He has co-authored over 20 scientific publications in international journals and conference proceedings, and he has been involved in many different national and European research projects in the networking area.
His current research interests include wireless networks, estimation of freeway vehicular traffic, pervasive communications and mobility management, energy-efficient networking.