ethernet alliance

Improving the Energy Efficiency of Ethernet-Connected: A Proposal for Proxying

Version 1.0 September 2007

Authors:

Bruce Nordman, Lawrence Berkeley National Laboratory

Ken Christensen, University of South Florida

ethernet alliance | p.o. box 200757 | austin, tx | 78720-0757 | usa
www.ethernetalliance.org

Executive Summary

Being connected to the Internet requires some active participation. When devices fail to do this, they "fall off the net" and applications break. Today, billions of dollars' worth of electricity are used to keep Ethernet (and other) connected devices fully powered on at all times only for the purpose of maintaining this connectivity. If not for the network connectivity, most of these devices could be asleep the majority of the time, with greater energy savings resulting. Saving energy can be achieved by the addition of a proxying capability, which is defined as "an entity that maintains full network presence for a sleeping device". This new proxying capability will enable existing PC power management features to be much more utilized, and can do so without requiring changes to existing network applications. This white paper describes the basic ideas behind proxying and how they could be implemented to achieve large world-wide energy savings.

Introduction

The Internet protocols were designed when there were relatively few devices connected to the Internet and these devices were in use most of the time. As such, the concept of power states is not present in the network architecture of the Internet. Power states would be useful to implement within devices— for example, the system sleep state supported by PC operating systems— are not defined externally to the devices. Today, literally hundreds of millions of devices are connected to the Internet via Ethernet links, with the potential to grow by one or two orders of magnitude. The majority of these devices are relatively idle most of the time. At this time, many of these devices are desktop and notebook PCs, but in the future it may be set-top boxes or other items yet to be developed. Billions of dollars of electricity are being spent to keep this equipment fully powered when no user is present and network access is sporadic or incidental. Saving this wasted energy can be done two ways:

- Redesigning network protocols and applications, or
- Encapsulating the intelligence for maintaining network presence in an entity other than the core of the networked devices.

The first option seems infeasible for any near-term timeframe. The second option is quite doable in the near future. Called "proxying" [1] it is defined as "an entity

that maintains full network presence for a sleeping product".
The remainder of this white paper covers the proxying concept in more detail, and

Potential for Energy Savings

A PC can and should be asleep if no one is actively using it and it is not delivering an ongoing service such as streaming audio or video. While there are a variety of reasons for the lack of using sleep modes, losing network connectivity appears to be the largest barrier and growing as an increasing number of applications rely on continuous connectivity. For desktop PCs in the U.S., most time spent while fully on meets these criteria so that proxying could save *more than half* the energy used by these products, since sleep power level is significantly less than idle power.

Studies [2] have found that most office desktop PCs are left on continuously and an increasing number of residential PCs are as well [3]. Estimates of the number of PCs in use are calculated from estimates of sales. An early estimate [4] put the savings potential for the U.S. at $0.8 to $2.7 billion/year of direct energy use. Since that time, electricity prices and the total number of PCs have gone up (increasing savings), though the percent of systems that are desktops has declined (decreasing savings). There are also potential savings from other Ethernet-connected devices such as printers and some set-top boxes. In the future one can expect many more consumer electronic devices to have IP connectivity and so be able to benefit from proxying functionality.

Network Protocols and Power State

The Internet was designed with the idea that devices have no network power state – they are either fully on and present on the network, or they are off the network and possibly powered off. Application and protocol state are maintained in the edge devices rather than in the core of the network. This principle has been critical to the success of the Internet and should be generally maintained. Ensuring that devices can enter low-power "sleep" states without compromising their network presence requires an entity other than the main device to maintain their network presence. This would appear to violate the principle of avoiding network state and intelligence in the network infrastructure itself. However, the key is to isolate the perturbation

of the network to well-defined parts near the edge to insulate the network from the change so that other devices see no change in network behavior from the sleeping device. Higher layer protocols may find it helpful to have knowledge of device power state. Such knowledge can be used to re-route requests to other devices that are not sleeping or to schedule activities in ways that maximize sleep time. Defining extensions to existing management protocols to communicate power state is an area of great future potential.

Network devices that are not in active use can be powered-down to save energy. This was first recognized in the early 1990s when energy costs for PCs became noticeable for large organizations. Methods were developed to remotely power-up devices. Remote wake-up was, and still is, necessary to be able to remotely access and/or manage devices.

Network Wake-up Events

A network wake-up event is a request sent as a network message to wake-up a sleeping device into a fully powered-on state. This capability is popularly known as Wake on LAN (WOL). A WOL message can be a specialized Magic Packet [6] or any packet that contains a bit pattern that matches a wake-up pattern loaded into an Ethernet NIC. A Magic Packet is a specialized packet that contains the hardware (Ethernet) address of the device repeated 16 times [5]. Packet patterns [6] that define wake-up packets for PCs running Microsoft Windows may include:

- NetBIOS over TCP/IP broadcast for computer name assigned to device;
- Address Resolution Protocol (ARP) broadcast for IP address of the device; or
- Any packet that contains the IP address of the device.

Magic Packet was developed in the early 1990s by IBM and AMD when IP networks using routers were not yet commonplace. It is difficult to send a Magic Packet across the Internet, so it is primarily useful on local subnets. Magic Packet also suffers from the lack of an industry standard and is not part of the Internet protocols. For example, Magic Packet is not part of TCP connection establishment. A Magic Packet must be sent by an application specifically designed for waking-up sleeping devices. Wake-up on pattern match has the potential for allowing devices to wake-up on standard Internet packets, and thus transparently using existing protocols and applications, and cross subnet boundaries. Unfortunately, due to overly generalized

packet patterns, wake-up on pattern often results in frequent wake-up for trivial events or even non-events, and/or a failure to wake-up when needed. Thus, a key issue is that devices wake-up too often or not often enough, resulting in a greatly reduced energy savings and/or loss of functionality, usually followed by a user disabling all power management features resulting in no energy savings.

Proxying for Ethernet and IP Networks

Reliable and standard wake-up is needed to bring powered-down devices back into a fully powered state when their resources are needed. However, the full resources of a device, such as a desktop PC with a powerful (and power hungry) processor and significant amounts of memory and storage, are not needed to handle many network messages destined for a system. A proxy is able to provide a solution to this concern.

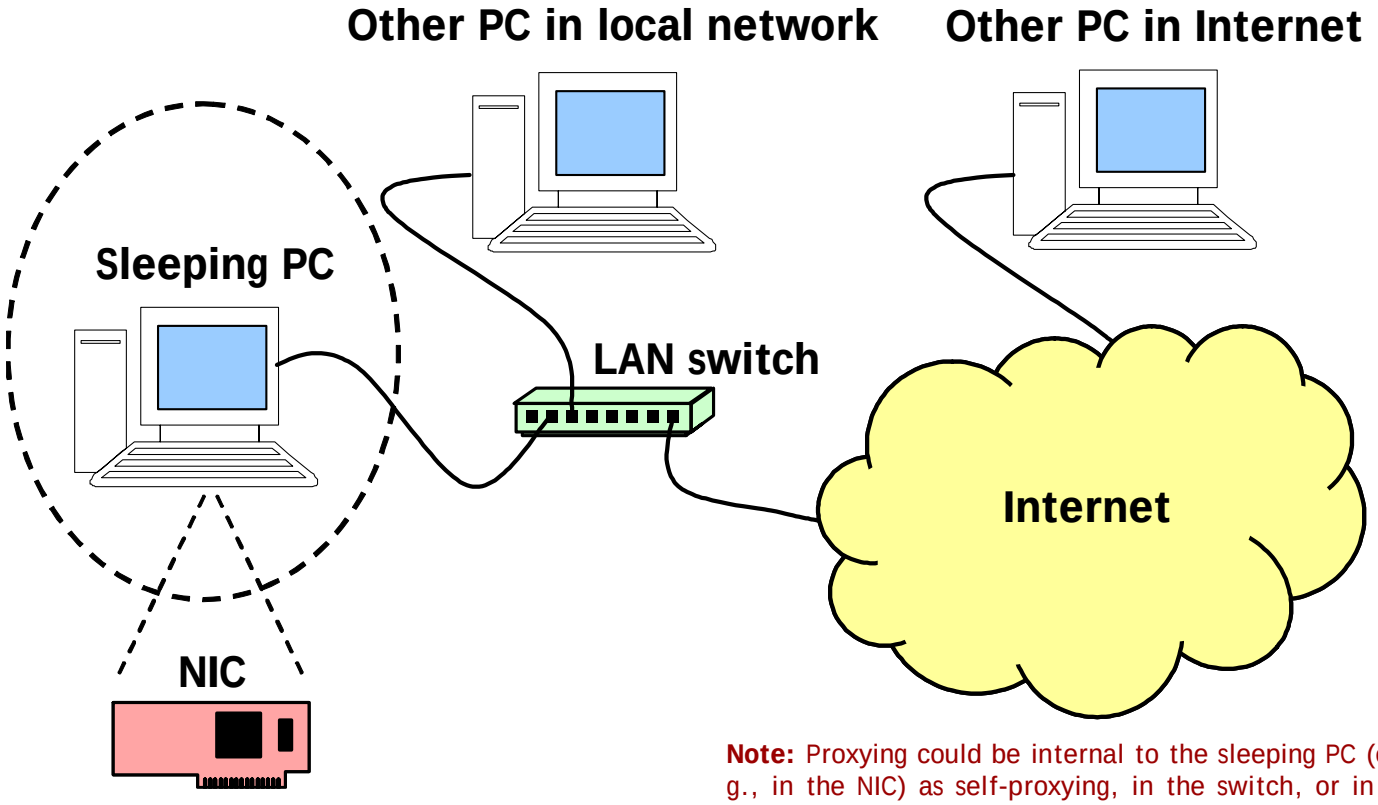
A proxy performs four basic functions: a) responding to routine requests; b) auto-generating routine replies; c) identifying when a wakeup is truly warranted; and d) ignoring all other packets. These functions are described in more detail later in this white paper. Figure 1 shows the possible placement of proxying functionality in a network. There are three basic approaches to implementing proxying:

- **Self proxying.** This puts the proxy functionality into hardware within the connected product itself, for example within the Network Interface (NIC). The key is to not require the power-intensive main processor, memory, and most buses to be active during sleep.

- **Switch proxying.** This puts the proxy functionality into the immediately adjacent network switch so that the end device itself need not be changed but no other devices on the network are aware of the device being asleep. Switches are assumed to be always fully on.

- **Third-party proxying.** This puts the proxy functionality somewhere else in the network other than the device or immediately adjacent switch. Key challenges are to ensure that packets from other devices on the Internet destined for the sleeping device make it to the proxy and that reliable wake-up mechanisms can be implemented.

The simplest case is self proxying, since only one device (under the control of a single operating system) is involved. No other products or protocols are aware of the device being asleep and so do not change their behavior in any way. Transferring state between the sleeping device and the proxy is simplest for self-proxying. Wireless networks have potential issues when a device moves to a new wireless LAN, particularly whether that event could or should cause a wake-up to allow the device to fully join.



Note: Proxying could be internal to the sleeping PC (e. g., in the NIC) as self-proxying, in the switch, or in a third-party PC or device in the local network or in the Internet

Figure 1— Possible Placement of Proxying Functionality

The next simplest case is switch proxying. It requires changing just two devices (switch and sleeping device), relies on existing wake-up mechanisms, and transference of network state is only between the two adjacent devices. Implementation of switch proxying in the wireless context introduces issues not present for wired connections which may be problematic for proxying. The mobility of the sleeping device raises issues for how to implement proxying.

Third-party proxying raises the most challenges in technical terms and is not consider in this white paper.

How Proxying Works

Before any proxying can occur, the device and proxy need to understand that each other both support the functionality. It is possible that some proxies may be more capable than others by proxying for more applications or protocols, so the degree of capability may need to be exchanged as well. PCs already exchange information with their network interfaces when systems are initialized. For products such as set-top boxes with fixed hardware, the capability can be built in so the exchange essentially takes place in advance. Figure 2 outlines the key functional steps for a proxy. The steps are:

1. The device determines that it is time to go to sleep (for example, based on inactivity).
2. Notice and state are passed to the proxy, and the device goes to sleep.
3. The proxy maintains full network presence.
4. The proxy determines when a packet requiring wakeup has arrived and signals the device to wakeup.
5. Once the device has fully woken up, state is passed back from the proxy to the device, and the device returns to normal network operation.

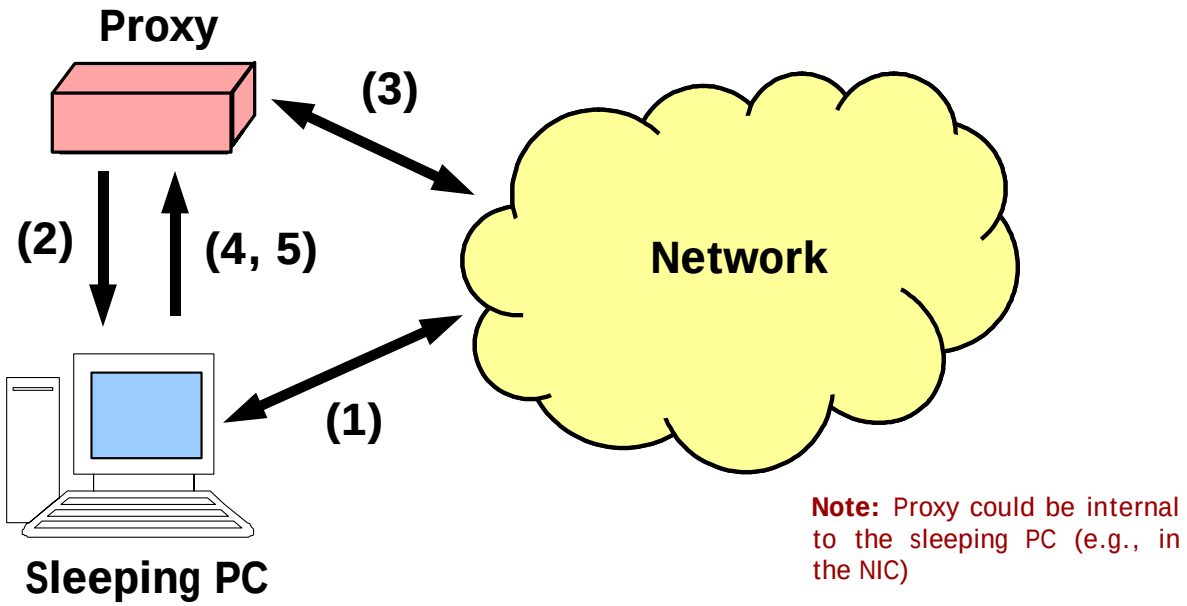


Figure 2— Operation of a Proxy

The sleeping device may wake itself from an internal timer, detected user activity, or a power supply condition. The proxy will trigger a wakeup when it notices a worthy event such as activity on the network of interest or perhaps user presence. The transfer of state back to the system could include the cause of wakeup. Each step is considered further below in more detail.

1. *The device determines that it is time to go to sleep.*
   Usually this is due to an inactivity timer expiring indicating that no user activity has occurred for a period of time, but can also be caused by a low battery condition, manual control, application request, or potentially at the direction of another device on the network (this is under the control of the operating system policy or a designated application).

2. *Notice and state are passed to the proxy. The device goes to sleep.*
   A notice to initiate proxying may be delivered through the existing interface (modified appropriately) between a processor and NIC, or through a special packet to the attached switch. The device will undertake its ordinary actions on entering sleep, including possibly reducing the link rate [7].

3. *The proxy maintains presence.*
   This has four components: respond to routine requests; auto-generate routine replies; identify when a wakeup is truly warranted; and ignore all other packets. The proxy may also maintain and update state (i.e. entries in an ARP cache) that will be transferred back to the device on wake-up (step 5 below).

4. *The proxy determines when a packet requiring wakeup has arrived and signals the system to wakeup.*
   The waking packet might be buffered in the proxy, as might additional packets that arrive during the waking period. Existing NICs have a wakeup mechanism internal to the PC to support WOL. For switch-based proxying, the wakeup could occur with WOL or some other mechanism possibly unique to the link type.

5. *Once the system has fully woken up, state is passed back from the proxy for the device to return to normal network operation.*
   The state transfer packet has much of the same information as the state transferred when proxying is initiated (in step 2). This happens before any buffered packets are sent, as the state information may affect how the system interprets the buffered packets. Whether the proxy continues to perform some or all of its functions during the wake-up process is not yet known.

The contents of this transferred state are not yet defined, but could include information about open TCP connections and proxying options. It also could include information about DHCP such as whether it is in use and if so, the lease time.

When a device wakes up, there is often a delay from when the wake-up signal occurs whether internally generated or from the network, and when the system is fully ready to receive and respond to network queries. For older PCs, this can be on the order of ten seconds or more, though on modern ones just a few seconds (Microsoft now specifies that Windows PCs shall wake up in less than two seconds). This time is very long by network standards, though not long enough to be a fundamental barrier. Network protocols generally have mechanisms to retry sending packets when they are not acknowledged because applications and higher-layer protocols on IP networks assume some degree of unreliability. Thus, accommodating the waking time is feasible for current applications and protocols. Future work may address mechanisms whereby a new type of packet responds back to the sending IP device to the effect that "I am waking up and expect to be responsive in X seconds so wait until then before sending more data".

An Existing Implementation of Proxying

Proxying to enable power management already exists for at least one specific protocol – Universal Plug and Play (UPnP). UPnP is an automatic configuration protocol for network devices. UPnP is well suited to residential use where a multitude of devices often enter and leave and need to work together when present without manual configuration. UPnP uses a fully distributed discovery protocol that requires all devices in a UPnP network to be fully powered-up at all times in order to respond to discovery messages. In August 2007 the UPnP Forum released the document for UPnP Low Power Architecture, Version 1.0 [8]. The objective of this architecture is to allow UPnP devices that implement power savings modes to be able to sleep and save energy, and still be discoverable by UPnP control points. A power management proxy service is key to this architecture.

Power management as defined for UPnP is not transparent. Changes to UPnP client functionality are needed. The architecture is specific to UPnP only. This solution to enabling power management for one specific protocol shows the feasibility of proxying and points the way to the need for a more general – and transparent – approach to proxying. This is the next step.

Next Steps to Realizing General Proxying

The key to developing a general proxying capability is a standard definition of "full network presence". This should apply to as broad range of Internet (IP) protocols and applications as possible – hence the idea of "general proxying". This will specifically include a list of packet types that may require routine reply, auto-generation, and wakeup, as well as the detailed steps each requires. The proxying standard could include a core or base set of functionality, along with optional additional capabilities (for example, to support particular applications or network environments). Also needed is the state information passed down to the proxy on going to sleep, as well as the state information passed up to the device on wakeup. Standard NIC-to-operating-system interfaces need to be extended to facilitate this. For switch-based proxying, a method needs to be defined to transfer state information. For all forms of proxying, security is a prime consideration. It is not clear that proxying introduces any security vulnerabilities greater than those of a device that itself remains fully powered-on at all times.

The packet type and response table should be part of the proxying standard. Possible standards bodies to tackle a general proxying standard are IETF, DMTF, and IEEE (MSC). The DMTF with its Alert Specification Format (ASF) standard [9] is a first step toward proxying. The ASF standard defines how ARP packets can be handled by an ASF-compliant NIC on behalf of a sleeping PC. A near-term task is to further find the best standard group with regards to topical suitability and interest. NIC-to-operating-system interface definitions already exist and could simply be amended, though a standard list of the state information passed back and forth is needed. For the time being, drafts of the proxying functionality will be hosted on the USF and/or LBNL web sites: http://www.csee.usf.edu/~christen/energy/main.html; and http://efficientnetworks.lbl.gov.

It should be noted that the ENERGY STAR program in its Tier 2 computer specification [10] is scheduled to require proxying functionality in ENERGY STAR-compliant desktop and notebook PCs beginning in 2009.

Conclusions

Defining general proxying functionality – then implementing it in hardware and software – can save billions of dollars of electricity per year. Perhaps no other source of energy savings in electronics is as large. General proxying is non-trivial, but no fundamental impediments to implementing it are apparent. Defining "full network presence" needs the close effort and scrutiny of network professionals from industry, academia, and standards organizations.

About the Authors

Bruce Nordman is a researcher with the Environmental Energy Technologies Department of Lawrence Berkeley National Laboratory. He has worked on energy efficiency in electronics at LBNL since the early 1990s, doing research for the EPA Energy Star program, California Energy Commission, and U.S. Department of Energy. You can contact Bruce at BNordman@LBL.gov, 510.486.7089, http://efficientnetworks.lbl.gov.

Ken Christensen is a Professor in the Department of Computer Science and Engineering at the University of South Florida. His interest in energy efficiency in computer networks goes back to the mid-1990s with work on Wake-On-LAN at IBM. His current research in energy efficiency of networks is funded by the National Science Foundation. The material in this white paper is based upon work supported by the National Science Foundation under Grant No 0520081. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author (s) and do not necessarily reflect the views of the National Science Foundation (NSF). You can contact Ken at christen@csee.usf.edu, 813.974.4761, http://www.csee.usf.edu/~christen/energy/main.html

Appendix A – Proxying FAQs

1) What is power management proxying?

A power management proxy, or just "proxy", is a low-power entity that maintains full network presence for a sleeping high-power device. A proxy covers (spoofs) for a sleeping device by generating and responding to routine network and application layer packets. A proxy wakes up the device it is covering for only when it cannot handle a valid request. Thus, a proxy enables a device to sleep – and thus save energy – whereas without proxying the device would not be able to do so.

2) What is the need for proxying?

Most energy used by desktop PCs occurs when no one is present and the system is mostly idle. An increasing number of applications rely on maintaining continuous network connectivity. Additional device types, such as set-top boxes, also require continuous connectivity. Proxying enables large energy savings in these products that no other solution can deliver.

3) What is the primary application for proxying?

AC-powered Internet-connected devices such as PCs, set-top boxes, printers, etc.

4) What are the expected operating cost savings achievable from proxying?

For an individual desktop PC, tens of dollars per year. For the U.S., billions of dollars per year. As an example, consider a PC that uses 65 W while on and 5 W while asleep, is on continuously, but only actively used 40 hours per week. Putting this PC to sleep the remainder of the time will save about 400 kWh/year, or $40/year at 10 cents/kWh. Fifty million of these would save $2 billion/year.

5) Is proxying technically feasible?

Yes. Many new Ethernet NICs have sufficient built-in processing capability today, as do network switches.

10) Can there be different levels of implementation of proxying?

Yes. There should be a core of basic proxying functionality that can guide product and application software designers, but specific products or applications may call for additional proxying capabilities on top of the core to maximize benefits.

11) Will proxying result in a user-perceptible performance impact?

User perception of the impact of proxying will vary from none to slight. In some cases, the wake-up period for a sleeping machine may impact responsiveness. This is an inherent trade-off in energy savings versus always-on responsiveness. The general trend among hardware and software vendors is for wake-up times of PCs and other equipment to become smaller.

12) Does proxying have to be a standard?

Yes. Without a standard, there is likely to be widespread user confusion about what products support what types of proxying, resulting in loss of much of the potential savings.

13) Is there any precedent or "competition" for proxying?

There is no alternative approach to get the full savings proxying can deliver. Improving existing WOL could deliver more savings than it does today, but would still deliver much less savings than proxying.

14) What are the key reference documents for proxying?

At present, this white paper, the cited references, and the evolving functionality description (see http://www.csee.usf.edu/~christen/energy/main.html and http://efficientnetworks.lbl.gov).

15) Why support an activity to standardize proxying?

The potential savings from widespread use of proxying are compelling, and attaining anything close to this level of potential savings will require a standard reference for core proxying functionality.

16) What makes a device relatively "idle"?

A device is considered idle when there is no user input and no ongoing activity such as media streaming.

17) What key assumptions drive savings estimates from proxying?

The key assumptions that drive the energy saving estimate are which products will benefit, how many there will be when proxying is deployed, the portion of year shiftable from on to sleep, and the power level difference between on and sleep.

18) Are there other benefits of proxying?

The proxying infrastructure could enable offloading of some routine network activity from devices to the proxy while the device is awake, providing a modest performance improvement. Also, the existence of proxying will lead to some people using it to leave their devices on (but asleep) rather than fully off and so gain functionality they lack at present.

19) Who are the key contacts for proxying?

The authors of this white paper.

References

[1] C. Gunaratne, K. Christensen, and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed," *International Journal of Network Management*, Vol. 15, No. 5, pp. 297-310, September/October 2005.

[2] J. Roberson, C. Webber, M. McWhinney, R. Brown, M. Pinckard, and J. Busch, "After-hours Power Status of Office Equipment and Inventory of Miscellaneous Plug-Load Equipment," *Technical Report LBNL-53729-Revised*, Lawrence Berkeley National Laboratory, 2004.

[3] K. Roth, K. McKenney, R. Ponoum, and C. Paetsch, "Residential Miscellaneous Electric Loads: Energy Consumption Characterization and Savings Potential," Prepared for the US. Department of Energy, July 2007. A key quote (p. 4-29) is that

"Interestingly, no one is using the PC during more than half of this time spent in active mode…"

[4] B. Nordman and R. Brown. "Networks: Tomorrow's big challenge for IT energy use," Woodrow Wilson Center Workshop on Environment and the Information Economy: Next Steps for Research, March 15, 2004.

[5] "Magic Packet Technology," *Publication #20213*, AMD, November 1995. URL: http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20213.pdf.

[6] "Power Management for Network Devices," Microsoft, 2001. URL: http://www.microsoft.com/whdc/archive/netpm.mspx.

[7] M. Bennett, K. Christensen, and B. Nordman "Improving the Energy Efficiency of Ethernet: Adaptive Link Rate Proposal," Ethernet Alliance White Paper, Version 1.0, July 15, 2006. URL: http://www.ethernetalliance.org/technology/white_papers/alr_v10.pdf. Also, IEEE 802.3 Energy Efficient Ethernet Study Group. URL: http://www.ieee802.org/3/eee_study/index.html.

[8] UPnP Low Power Architecture, Version 1.0, UPnP Forum, August 27, 2007. URL: http://www.upnp.org/specs/lp.asp.

[9] Alert Standard Format (ASF) Specification, v2.0, DSP0136, Distributed Management Task Force, Inc., April 23, 2003. URL: http://www.dmtf.org/standards/asf/.

[10] U.S. EPA, "ENERGY STAR® Program Requirements for Computers", September 2006. URL: http://www.energystar.gov/index.cfm?c=archives.computer_spec.