Design and Implementation of a Power Management Proxy for Universal Plug and Play

Jakob Klamra, Martin Olsson
Dept. of Communication Systems
Lund Institute of Technology
Lund, Sweden
jakob.klamra@gmail.com,
molsson@hotmail.com

Ken Christensen
Dept. of Computer Science and Eng.
University of South Florida
Tampa, Florida, USA 33620
christen@cse.usf.edu

Bruce Nordman
Energy Analysis Department
Lawrence Berkeley National Lab
Berkeley, CA, USA 94720
bnordman@lbl.gov

***Abstract*– Universal Plug and Play (UPnP) is an automatic configuration protocol for network devices. A key component in UPnP is the Simple Service Discovery Protocol (SSDP). SSDP is a fully distributed discovery protocol and requires all devices in a UPnP network to be fully powered-up at all times in order to send periodic advertisements and respond to discovery messages. In this paper, the design and implementation of a UPnP power management proxy is described. This proxy makes it possible for UPnP devices to enter and remain in a low-power sleep state and thus reduce the energy used by the device. Wake-on-LAN is used by the proxy to wake up sleeping UPnP devices only when their services are needed. The proxy is completely backwards compatible with existing UPnP standards. The estimated economic savings in reduced electricity use is between $125 and $312 million per year in the US alone, if this approach was to be widely adopted.**

I. INTRODUCTION

The amount of energy used by networked electronic devices is growing rapidly. In 1999 electricity use by office and network equipment was about 2% of the total electricity consumed in the USA corresponding to $6 billion in energy costs and the emission of more than 50 million metric tons of carbon dioxide per year [1]. This quantity has increased since 1999. Within the typical US residence, electricity use by electronic equipment when in a low-power mode (i.e., not in active use) is now about 1000 kWh/year (approximately $80 per year) and is greater than the energy used by a refrigerator [2]. Research into improving the energy efficiency of electronic devices – equipment that is almost always connected to a network – is of great economic and environmental significance [3], [4]. Energy efficiency is also of concern for mobile devices where battery lifetime is often the limiting design constraint and impairs functionality.

Dynamic power management methods address the problem of energy use of electronic devices by putting devices into low-power sleep states when they are not in active use. A timer can be used to detect user inactivity and predict that inactivity will likely continue. Upon expiration of the inactivity timer, the device will enter sleep mode. On detection of user activity, or receipt of a Wake-on-LAN (WOL) packet, the device resumes a fully-powered state. The time to power up and down between sleep and fully operational states is non-negligible and thus powering down should only take place for long periods of inactivity.

Increasingly, network protocols have been found to *induce* energy use in devices. That is, network protocols are forcing devices to remain fully powered-up continuously, even when not in active use, to respond to protocol messages. One example of this is the Universal Plug and Play (UPnP) automatic configuration protocol and its underlying Simple Service Discovery Protocol (SSDP) [5]. UPnP is a standard protocol intended to extend the PC peripheral plug-and-play concept to networks [6]. The network medium can be wired (e.g., Ethernet) or wireless (e.g., WiFi). UPnP has the potential to be a widely deployed protocol in residential networks. Microsoft, Intel, and Nokia are key supporters of UPnP. In a UPnP network – that is, a network where all devices execute a UPnP protocol implementation – a new device brought into a network can automatically discover available services in other devices. The discovered services can then be used (i.e., are automatically configured) by the new device. It is the underlying SSDP protocol that requires all devices to be fully powered-up in order to send and respond to SSDP messages at all times. Thus a device that enters a low-power sleep state and ignores most network traffic is considered disconnected from the network. When configuring a UPnP network, it is required that power management is disabled in all devices. In this paper, we address how to allow UPnP devices to power down to a low-power sleep state and still be fully discoverable by SSDP (i.e., how to enable power management in UPnP devices). We approach this problem with the design and implementation of a proxy to "spoof" for powered-down devices and wake them up when their services are required.

The remainder of this paper is organized as follows. Section II describes the design of a UPnP power management proxy. Section III describes implementation and evaluation of the proxy. In Section IV possible energy savings are estimated. Section V briefly describes related work. Section VI is the summary and outlines future work.

II. DESIGN OF THE POWER MANAGEMENT PROXY

In this section we present an overview of the SSDP protocol in UPnP and describe the design of a power management proxy. Two versions of a UPnP power management proxy are considered – an invisible proxy and a cooperating proxy.

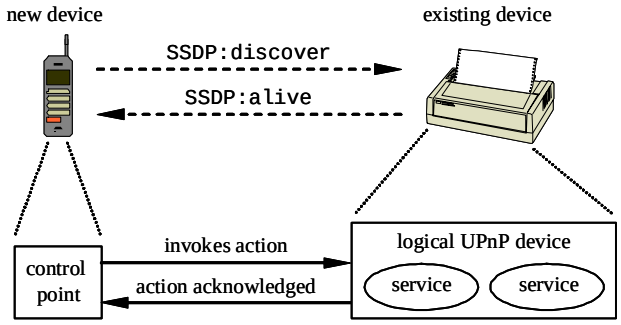

Fig. 1. UPnP devices showing services and control points

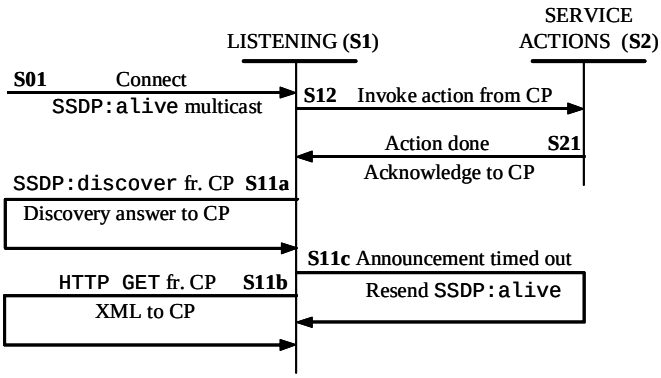

Fig. 2. Simplified FSM for a UPnP service

A. *Overview of discovery in UPnP*

UPnP is fundamentally a client/server protocol where devices can serve as a client, server, or as both. UPnP uses the concepts of *device*, *service*, and *control point*. A device is a physical entity such as a mobile phone, printer, or PC. Within devices are logical services and control points. A printer may contain services for printing and faxing. A mobile phone may contain a control point for invoking the services in a printer (e.g., to print pictures stored in the mobile phone that can be sent to the printer via Bluetooth). Devices with services must advertise their presence and do so via periodic, multicast SSDP *presence announcements* (`SSDP:alive`). Information contained in an `SSDP:alive` is only valid a limited time, and must therefore be periodically resent. This is to determine which devices may have unexpectedly left the network. Control points need to discover services in devices and do so via multicast SSDP *discovery requests* (`SSDP:discover`). Discovery messages are typically sent out when the control point connects to the network. Fig. 1 shows two devices, one with a control point and one with a service, and the key SSDP packet flows between the devices. Fig. 2 shows a simplified FSM (the complete FSM is in [7]) for a UPnP service. Transition S11a is the action of receiving a discovery packet from a control point and replying with a discovery acknowledgement packet. Transition S11b shows the action of receiving an `HTTP GET` from a control point and replying with an XML scheme describing service capabilities.

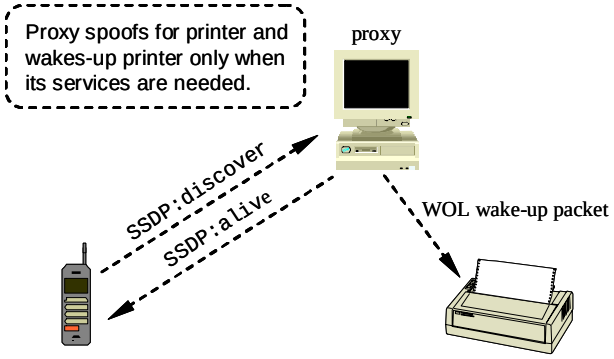

Fig. 3. UPnP power management proxy

Transition S11c shows the action of updating timed out information about the service. Finally, transitions S12 and S21 show a control point using the service.

It is transitions S11a and S11c in Fig. 2 that are of primary concern. If a device enters a sleep state, the discovery message cannot be replied to and periodic presence announcement messages cannot be sent. Could the discovery message be used to trigger a wake up of a sleeping device? This is infeasible because existing Ethernet and WiFi NICs cannot recognize a discovery message as a trigger for wake up. Existing NICs can, however, recognize a WOL packet for wake up. Existing NICs would need to be redesigned and all existing UPnP devices replaced if a discovery-triggered wake up was to be used In addition, if discovery messages were to be used to trigger a wake up, the device would be awake much more than is necessary thus reducing the amount of energy saved. Finally, wake-up for discovery is insufficient – periodic presence announcement messages also need to be sent (again, requiring a device to be fully powered). An alternative solution is to have a proxy reply on behalf of all sleeping devices. This is the approach investigated in this paper.

B. *Design of an invisible proxy*

A power management proxy is an always-on service in a device that does not enter a low-power sleep state. An invisible power management proxy requires no changes to the devices for which it is proxying. Fig. 3 shows how such a proxy can spoof for, or act on behalf of, a device in a sleep state. The proxy spoofs periodic `SSDP:alive` messages and answers `SSDP:discover` messages destined for the sleeping device. If there is an incoming request that the proxy cannot answer it will wake up the sleeping device using a standard wake up mechanism (e.g., sending a WOL packet).

The invisible proxy does not communicate directly with any device nor announce its presence in the network. This makes the proxy invisible to all other devices in the network. Fig. 4 shows the design of the invisible proxy (the complete design is in [7]). Since the proxy will not be notified of a device entering a sleep state it must monitor the traffic on the network and calculate the time since the last activity from all devices. If there has been no activity from a certain device after a threshold time, the proxy will start proxying for the device. This is shown in transition S12 in Fig. 4. While in

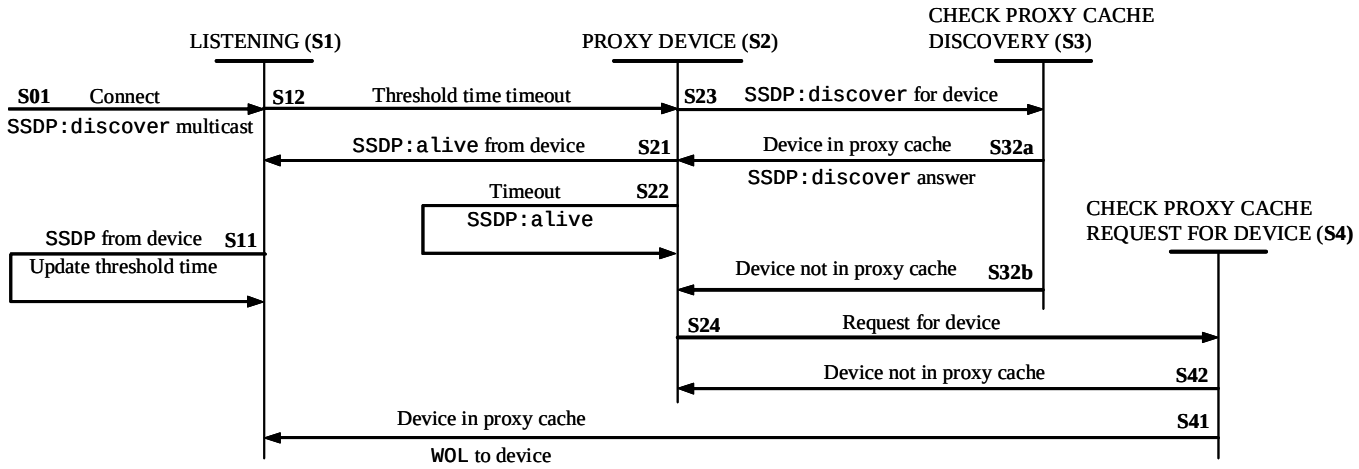

Fig. 4. FSM for an invisible UPnP power management proxy

state S2 the proxy will send periodic `SSDP:alive` messages as seen in transition S22. The proxy will also reply to incoming `SSDP:discover` messages destined for the sleeping device in transitions S23 and S32a. If the proxy receives a request for a sleeping device, it will wake up the device using a WOL packet in transitions S24 and S41.

For a proxy to work in a switched network – where no one device can "see" all the traffic in the network – a method of redirecting packets to the proxy can be used. When the proxy takes over for a presumed sleeping device, it can send an ARP reply to the entire network associating the proxy's MAC address with the IP address of the sleeping device. When the device is detected by the proxy as being powered-up again, the proxy sends another ARP reply to re-associate the MAC address of the device to the IP address of the device

C. Design of a cooperating proxy

The invisible proxy cannot verify the power state of a device. If a device were to leave the network unexpectedly, the invisible proxy will "see" this as the device going to sleep and will begin acting for the now non-existent device. This error will not be discovered until the proxy tries to wake up the device. To correct this shortcoming, the cooperating proxy was designed.

The cooperating proxy announces its presence on the network and communicates directly with all devices in the network. To enable this communication each power managed UPnP device must implement a new power management service. When a device enters sleep mode it will change the state of its power management service, automatically notifying the proxy using the GENA protocol. The cooperating proxy will then start acting for the device. When the sleeping device wakes up, the state of its power management service will change again and the proxy will be notified. Using this design the risk of a proxy acting for disconnected devices is minimized. The complete design of the cooperating proxy can be found in [7]. The design entails a modification of the FSM of Fig. 4 adding three new states and ten new transitions.

D. Design trade-offs and open issues

Neither of the two proxy designs can determine if a sleeping device has left the network resulting in false proxying. Another issue is how to select a proxy if more than one device in a network is capable of proxying. An election procedure could be implemented. If more than one device is capable of being a proxy, a means of backing-up each other could be designed. The cooperating proxy design is already enabled for proxy-to-proxy communication and is the logical starting point for designs to implement failure detection and recovery.

III. Implementation and Evaluation of the Proxy

The invisible and cooperating proxies were implemented as Microsoft Windows applications hosted in a standard PC. The proxy implementations were written in the C language using the Bloodshed Dev-CPP environment. The NetWib library was used for the communications interface. The key components in the implementation are caches and threads. Both proxies have a device and proxy cache. The device cache contains basic information about all other devices in the network. This information is required to construct answers to `SSDP:discover` messages. The proxy cache contains the same information, but only about those devices that the proxy is acting for at a given moment.

The implementation of the invisible proxy contains a main program and two threads – the proxy cache update thread and the notification thread as shown in Fig. 5. The main program listens to all traffic on the network and processes all packets. The main program adds new devices that connect to the network to the cache and removes devices that leave the network from the cache. The proxy update thread calculates the time since the last detected activity from a device. If the threshold time value is reached, the information about the device is copied from the device cache to the proxy cache. The notification thread continually checks whether it is time to spoof an `SSDP:alive` for a device in the proxy cache.

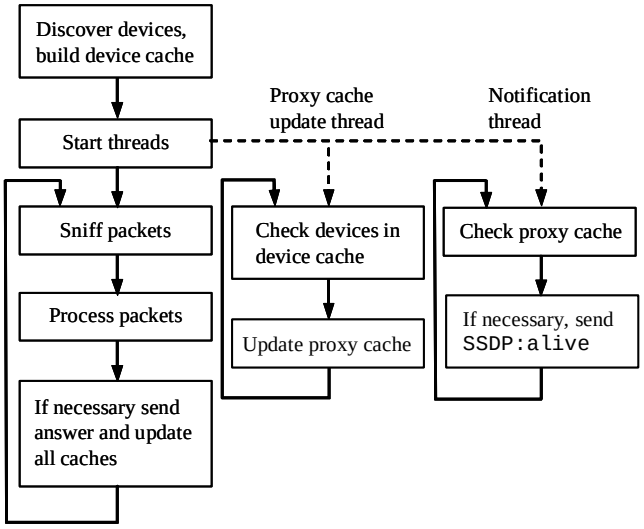

Fig. 5. Implementation of the invisible proxy

The implementation of the cooperating proxy is based on the invisible proxy. The main difference is the removal of the proxy update thread and the addition of a read event thread and an update device cache thread. The read event thread keeps track of incoming GENA events in the case of a device entering or leaving sleep mode. The thread will either copy information from the device cache to the proxy cache or remove information from the proxy cache, depending on the event. The update device cache thread removes information from the device cache if the information has timed out (which means a device has unexpectedly left the network).

Both proxy implementations have been tested with a variety of UPnP devices. Both proxy implementations allowed devices connected to the network to enter a sleep state without breaking any existing UPnP functionality. See [7] for a description of the validation and verification.

IV. Expected Energy Savings

If the UPnP proxy was deployed in future UPnP networks, what would the electricity savings be? It is unknown what the full penetration of UPnP will be in the next 5 to 10 years. It is equally unknown what new networked devices will be developed. However, a rough electricity savings estimate can be made by using stock estimates (for network connected devices in 2008) for notebook and desktop computers, and laser printers (which can be multi-function devices) [8]. An estimate based on only these devices will necessarily be conservative given that mobile and specialized UPnP devices (e.g., IP radio, home appliances, etc.) are entirely omitted, as are set-top boxes, media storage servers, and other power-intensive products. Table 1 shows the number of devices in millions, the power ratings for fully powered-up ("on") and sleeping, and the number of hours per week of sleep with or without a proxy [8]. For printers, the "on" power is the sleep mode of the printer in which its processor remains powered-up; the "sleep" line is a reduced sleep mode in which the processor can actually go to sleep. From Table 1, the yearly

TABLE 1
ESTIMATED ENERGEY SAVINGS ASSUMPTIONS

	Notebook	Desktop	Laser
Number of devices	42.5	84.8	11.3
Power in "on" (W)	22	82	15
Power in "sleep" (W)	3	6	5
On w/out proxy (hrs/wk)	56	56	56
On w/ proxy (hrs/wk)	19	15	1

energy savings are calculated in Wh. We estimate 10% (low estimate) to 25% (high estimate) of these devices will be UPnP enabled and using the proxy system. At 8 cents per kWh, the total savings range from $125 million to $312 million per year if UPnP networks had a power management proxy. Clearly, this estimate has many factors that could cause variation, but we believe this estimate to be conservative for the reasons described earlier in this paper.

V. Related Work

Significant research has been done in the area of power management focusing on predictive time-out schemes, frequency-voltage scaling, and other device-specific methods. Little work has been done focusing on power management from a systems (or network-wide) perspective. Proxying for power management was first proposed by Christensen and Gulledge [9] to handle general IP protocols such as ICMP ping and ARP on behalf of sleeping PCs on a shared-medium Ethernet. Recent work by Rosu et al. [10] has focused on using an HTTP proxy to shape incoming traffic to create longer idle periods, thus enabling wireless devices behind the proxy to sleep for longer periods of time. Kejariwal et al. [11] have proposed proxying as method of splitting functionality (in this case for computationally complex tasks) between energy-constrained handheld devices and a mains-powered proxy to reduce energy use of the handheld. The focus is on conserving handheld battery energy use and not on overall reduction of energy use.

The UPnP Forum [6] has a Low Power Working Committee (LPWC) addressing power management for UPnP. The LPWC is drafting a proposal for a power management proxy. The draft proposal is different from our proxy in that it requires significant changes to SSDP and other underlying UPnP protocols. Our proxy is of interest to the power management research community because it is fully backwards compatible with existing UPnP protocols, and an implementation now exists to allow further experimentation and serve as a base for a deployable product. The first two authors (Klamra and Olsson) are members of the UPnP Forum and the LPWC, and have contributed ideas to the Forum's power management effort.

VI. Summary and Future Work

Bringing power management into the network context is a new focus of our work. Proxying is a useful method for allowing devices to enter a low-power sleep state while maintaining a virtual presence in the network (via an always powered-on proxy). The feasibility of proxying for UPnP has

been shown by designing and implementing a fully backwards compatible power management proxy. The proxy implementation is freely available from the authors [7].

Future work is in several areas. There are open problems in how a proxy should be selected (e.g., if there are multiple devices that are proxy-capable) and how proxied devices should monitor for the possible failure of a proxy. For devices where the network interface is a small fraction of the overall device power consumption (e.g., as is the case for desktop PCs), the possibility of adding fully-distributed proxying on WiFi and Ethernet NICs can be explored – this "SmartNIC" concept is described in [4]. The general direction of bringing power management "outside of the box" has great potential for reducing the energy use of electronic equipment and requires further research. The benefits are to both the environment in overall reduced energy use and to enabling battery-powered handheld devices to increase their functionality without increasing their use of battery power.

ACKNOWLEDGMENTS

The first two authors (Klamra and Olsson) thank the support of *Anna Whitlocks Minnesfond*, *Carl Erik Levins Stiftelse*, *Stiftelsen Carl Swartz Minnesfond* and *Stiftelsen Sigfrid och Walborg Nordkvist* for travel funding to the USA for completion of this project.

REFERENCES

[1] 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," *Technical Report LBNL-45917*, Lawrence Berkeley National Laboratory, February 2001.

[2] B. Nordman, "Developing and Testing Low Power Mode Measurement Methods", #500-04-057, LBNL, prepared for the California Energy Commission, Public Interest Energy Research Program, September 2004. URL: http://www.energy.ca.gov/pier/final_project_reports/500-04-057.html.

[3] M. Gupta and S. Singh, "Greening of the Internet," *Proceedings of ACM SIGCOMM*, pp. 19-26, August 2003.

[4] K. Christensen, B. Nordman, and R. Brown, "Power Management in Networked Devices," (Communications column) *IEEE Computer*, Vol. 37, No. 8, pp. 91-93, August 2004.

[5] Y. Goland, T. Cai, P. Leach, and Y. Gu, "Simple Service Discovery Protocol/1.0 Operating without an Arbiter," draft-cai-ssdp-v1-03.txt, IETF Draft, October 28, 1999.

[6] *UPnP Forum*, 2005. URL: http://upnp.org.

[7] J. Klamra and M. Olsson, "Design and Evaluation of Power Management Support for UPnP Devices," Masters Thesis, Department of Communication Systems, Lund Institute of Technology, June 10, 2005. URL: http://www.csee.usf.edu/~christen/upnp/.

[8] B. Nordman and A. Meier, *Energy Consumption of Home Information Technology*, Lawrence Berkeley National Laboratory, LBNL-53500, July 2004.

[9] K. Christensen and F. Gulledge, "Enabling Power Management for Network-Attached Computers," *International Journal of Network Management*, Vol. 8, No. 2, pp. 120-130, March-April 1998.

[10] M. Rosu, C. Olsen, L. Luo, and C. Narayanaswami, "The Power-Aware Streaming Proxy Architecture," *Proceedings of BROADNETS*, October 2004.

[11] A. Kejariwal, S. Gupta, A. Nicolau, N. Dutt, and R. Gupta, "Proxy-Based Task Partitioning of Watermarking Algorithms for Reducing Energy Consumption in Mobile Devices," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 556-561, June 2004.