18

Energy-Efficient Protocol Design

Giuseppe Anastasi[1], Simone Brienza[1], Giuseppe Lo Re[2] and Marco Ortolani[2]
[1]*Department of Information Engineering, University of Pisa, Pisa, Italy*
[2]*DICGIM, University of Palermo, Palermo, Italy*

18.1 Introduction

In developed countries, the total energy consumed by the Internet accounts for approximately 2–3% of the overall worldwide energy consumption [1, 2]. Although this percentage is not so high, its absolute value is very remarkable and has followed an increasing trend over the years [3]. More important, it has been estimated that a large fraction of the overall energy consumed by the Internet is wasted due to an inefficient utilization of infrastructure and user equipment [4]. Hence, significant energy savings could be achieved through appropriate power management strategies. This has stimulated the interest and efforts of the research community. So far, most of the research projects and activities have been driven by telcos and Internet Service Providers (ISPs) and, thus, they have been aimed at reducing the energy consumption mainly in the Internet core (i.e., at routers) and at data centers [5]. Less attention has been devoted to reducing the energy consumption of *edge devices* at user premises (i.e., PCs, printers, IP phones, displays).

This chapter focuses on solutions for optimizing the energy consumption of PCs and other user equipment connected to the Internet. These edge devices account for the major fraction of the overall Internet-related energy consumption [6]. Even if the power consumed by a single edge device is limited (e.g., about 100 W for desktop PCs and about 20 W for notebooks) – because of their large number and utilization time – the total consumed energy can be huge. Moreover, edge devices are typically used with little or no attention to the energy problem. For example, many PCs are left on, even overnight and during the weekend, because of laziness, carelessness, or to maintain network connectivity (e.g., for peer-to-peer file sharing). In addition, users often do not use energy saving policies (e.g., automatic hibernation after a certain period of inactivity). Other edge devices, such as printers, IP phones, and displays, are typically kept *always on*, especially in offices and public buildings. Every year, the estimated

Green Communications: Principles, Concepts and Practice, First Edition.
Edited by Konstantinos Samdanis, Peter Rost, Andreas Maeder, Michela Meo and Christos Verikoukis.

overall energy consumption due to edge devices in United States is in the order of tens of TWh, causing an expense of billions of dollars. The need for specific solutions to the problem is, thus, quite apparent.

In the next sections, we consider the main approaches to power management of PCs and other user equipment connected to the Internet. Specifically, we introduce a general taxonomy to classify the proposed solutions. Then, according to the introduced taxonomy, we survey the main proposals presented in the literature. Obviously, most of the proposed solutions refer to PCs. However, some of them could be extended to other edge devices as well.

18.2 General Approaches to Power Management of Edge Devices

In order to identify possible approaches to energy efficiency in edge devices, it is necessary to determine preliminarily the cause of energy waste. One of the fundamental causes is the fact that many users leave their PC always on (especially in their workplace), due to laziness and/or carelessness. This clearly emerges, for example, in the PC energy report [7] about the energy consumption of PCs used at work, issued by the UK National Energy Foundation. This report highlights that about 21% of PCs used at work are almost never turned off (during nights and weekends), thus resulting in a waste of energy equal to approximately 1.5 TWh per year (corresponding to 700,000 tons of CO_2). In order to reduce this energy waste due to laziness and carelessness, PCs and other edge devices could be forcedly turned off at a certain time, employing common solutions, such as Nightwatchman [7], which is already used in many environments.

There are cases, however, in which PCs are deliberately left on for the execution of certain network activities, such as remote connection or P2P file sharing. Since the PC is used for the execution of that particular application only for a limited time interval, most of the energy consumed to maintain connectivity could be saved by introducing appropriate mechanisms for power management [8]. However, to be effective these mechanisms should save energy without introducing a significant degradation in performance. Some studies related to network traffic [9, 10] have shown that PCs (and other edge devices) experience long idle periods, during which they might be turned off or placed in *sleep* state, thus resulting in significant energy savings. Specifically, we need mechanisms that can allow a PC to sleep during idle periods and to resume promptly whenever an external packet is received or the user wants to use her/his PC, so as to minimize the impact on the system responsiveness.

Figure 18.1 shows the possible approaches aimed at reducing the energy consumption of edge devices (with particular reference to PCs) by eliminating wastes during idle time. The approach based on *on-demand wake-up* consists in putting the PC in *sleep* state during periods of inactivity and waking it up, later, by means of a special message called *Magic Packet* [11]. Conversely, *proxying* can be used to allow the PC to be in *sleep* state during periods of inactivity and to interact, at the same time, with any remote *host* through the network. This technique allows to delegate to an entity, called *proxy*, the management of interactions with the network. The two mechanisms can also be combined. In this case, when the *proxy* is not able to handle the request received through the network, it wakes up the PC by sending a Magic Packet and passes the request. Thus, communications are handled in a completely transparent way to the external network (and users). The approach known as *context-aware power management*, instead, exploits some context information (for instance, the presence/absence of the user) for a finer grained power management. This approach can be used not only for PCs, but for

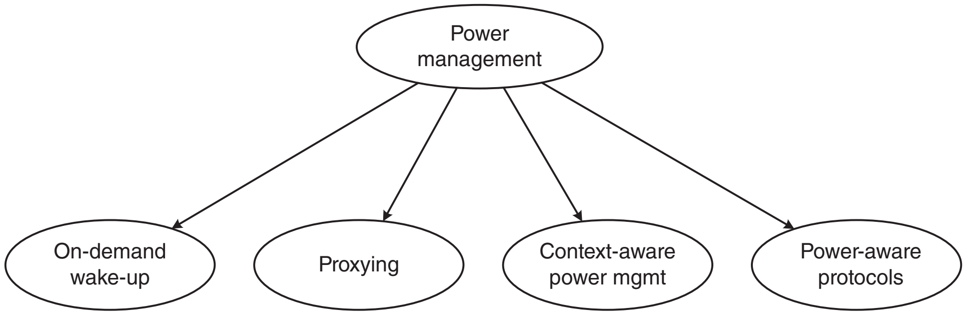

Figure 18.1 General approaches to power management of Internet edge devices

other edge devices as well. For example, displays used for the diffusion of information can be turned off when no one is in the nearby area. Finally, the use of *power-aware* protocols and applications is yet another approach to save power for edge devices.

In the next sections, the above-mentioned approaches and their potential to save energy are analyzed and explored in detail. It is important to emphasize that these approaches are not necessarily alternative but may also be used jointly.

18.3 Remotely Controlled Activation and Deactivation

Remotely controlled activation and deactivation is essentially based on the *Wake-on-LAN* (WoL) mechanism, widely used in Ethernet networks, which allows a PC in *sleep* state to be awakened – remotely – by sending a special message called *Magic Packet* [11]. The *Magic Packet* is typically sent from the same LAN of the target PC; however, it could also be sent by any device on the Internet. The WoL mechanism requires that part of the Ethernet network interface card (NIC) remains always active. Therefore, the PC cannot be disconnected from the power source or should use an alternative source of supply (i.e., a battery). The NIC component that remains active introduces, of course, a standby power consumption that is much smaller than the power consumed by the PC in the active mode.

The Magic Packet is a particular MAC frame that contains 16 repetitions of the MAC address of the target PC. It is usually sent as a UDP message destined to a specific port (9). After receiving a Magic Packet, the WoL component of the NIC wakes up the PC. There are two ways to transmit a Magic Packet over the network. It can be sent to the broadcast address of the subnet of the target PC (*subnet-directed broadcast*), or directly to the target PC (*unicast wake-up packet*).

The former case is the most common one since *subnet-directed broadcast* was the original transmission method for sending wake-up packets. With this technique, the Magic Packet is received by all the NICs in the network; however, it is discarded by all the NICs but the one whose MAC address matches the specified address. The main drawback is that, since Magic Packets are sent to a broadcast address, typically, they are not forwarded by routers. However, this limitation can be easily overcome in several ways, allowing a PC to be woken up by any computer on the Internet. This can be achieved, for instance, by configuring routers in such a way to allow them to forward Magic Packets. This solution, however, makes the network vulnerable to *DDoS attacks* (e.g., *Smurf Attacks*), that is, a malicious user could send a large amount of *ICMP* packets in broadcast, causing a remarkable response traffic. Another way is using a *virtual private network* (VPN) so that the remote computer appears to be a member of the same LAN as the sleeping PC.

The alternative solution is sending *unicast wake-up packets*. Since they are directed to the target IP address, they are routed through the Internet like regular datagrams. Nevertheless, this approach may not be compatible with all NICs, especially with oldest ones. In addition, such packets will not be delivered to PCs that have changed their IP address (e.g., via dynamic host configuration protocol (*DHCP*)) or whose address is no longer present in the *ARP cache* of the router.

Several extensions to the basic WoL mechanism described above have been proposed. For example, some Intel NICs allow several options, namely, *Wake on Directed Packet, Wake on Magic Packet, Wake on Magic Packet from power-off state,* and *Wake on Link*. In particular, *Wake on Directed Packet* [12] is an extension that makes the wake-on-demand mechanism much more flexible. Basically, a sleeping PC can be woken up by any packet directed to it, for example, by the request to open a TCP connection. Obviously, spurious wake-ups may occur with such mechanism, thus resulting in energy waste. Moreover, a PC consumes much more energy during the wake-up transition than during normal operating conditions. Therefore, spurious wake-ups should be avoided by filtering wake-up requests (see also Section 18.4).

Several power management systems for large-scale distributed systems have been proposed that make use of the wake on-demand mechanisms. *Polisave* [13] is a client–server system that allows to schedule actions for PCs associated with the service. In order to avoid energy waste, it is possible to specify the time when a client PC must be turned on/off, or must go into *standby* or *hibernation* state (i.e., the energy states defined in the *ACPI standard* [14]). In Polisave the client PC periodically queries the server in order to find out if there are actions scheduled for it. If a shutdown (or hibernation) is planned, the PC turns off. Conversely, if a PC is scheduled to be switched on, the server sends a Magic Packet to the NIC of the target PC.

Gicomp [15] is another similar system that allows to install/modify power management policies on controlled PCs. In particular, it allows to define the time to dim and turn off the display, the disk spin-down time-out, the suspend time-out, the hibernate time-out, and other options, according to the features offered by the operating system. Like Polisave, it follows a client–server paradigm. Clients and server communicate through the *XMPP protocol*, which guarantees confidentiality and authentication.

Both *Polisave* and *Gicomp* are able to work in the presence of network address translation (*NAT*) servers and *Firewalls*. They allow users to remotely control (through a web interface) all PCs associated with their personal account. PCs can be turned off, suspended, or turned on (through WoL). Finally, *Gicomp* solves the problem of Magic Packets' routing, using a specific *Waker* in each served IP subnet. *Wakers* are proxies, acting on behalf of the main server, that take care of waking up PCs on their IP subnet.

The on-demand wake-up technique has some limitations. First, it can be used only when the PC has a NIC with WoL support, that is, an Ethernet card. It also requires a proper configuration of both *BIOS* and operating system. It also suffers from security limitations. An attacker could turn on a sleeping PC through WoL, provided that she/he is on the same IP subnet of the sleeping PC. This is because the wake-up procedure does not require authentication and the content of the magic packet is transmitted as plaintext. To mitigate this problem, some NICs allow to insert a password in the Magic Packet, in addition to the MAC address.

However, this method can be easily overcome by sniffing the network traffic, as Magic Packets are not encoded. For these reasons, some PCs have an improved chipset to provide security for WoL. For instance, *Intel AMT* (a component of *Intel vPro technology*) supports *transport layer security* (TLS) encryption in order to secure an *out-of-band* communication tunnel for remote management commands such as WoL [16, 17].

18.4 Proxying

In this section, we analyze solutions on the basis of the use of a *proxy*. A proxy is an entity capable of responding to requests coming from the network on behalf of a sleeping device (e.g., a PC). The idea of using a proxy for energy conservation is not new. Indeed, proxy-based architectures have been proposed to guarantee an energy-efficient Internet access from mobile devices [18, 19]. However, in that case, the proxy architecture was designed to support a mobile device running standard client–server applications [19]. More recently, the idea of a proxy-based architecture has been extended to implement energy-aware solutions in the Internet [20]. In this case, the proxy acts on behalf of a host to respond to minimal network interactions and wakes up the host if needed.

A power management proxy works as shown in Figure 18.2. Initially, the edge device (e.g., a PC) has to associate with the proxy. Then, when the device is in sleep state and a request arrives from the network, there are two possible options. If the proxy is able to manage the received request by itself, it immediately serves it on behalf of the device, which can thus remain in sleep state. Otherwise, the proxy sends a wake-up message to the device and, then, forwards the received request to it.

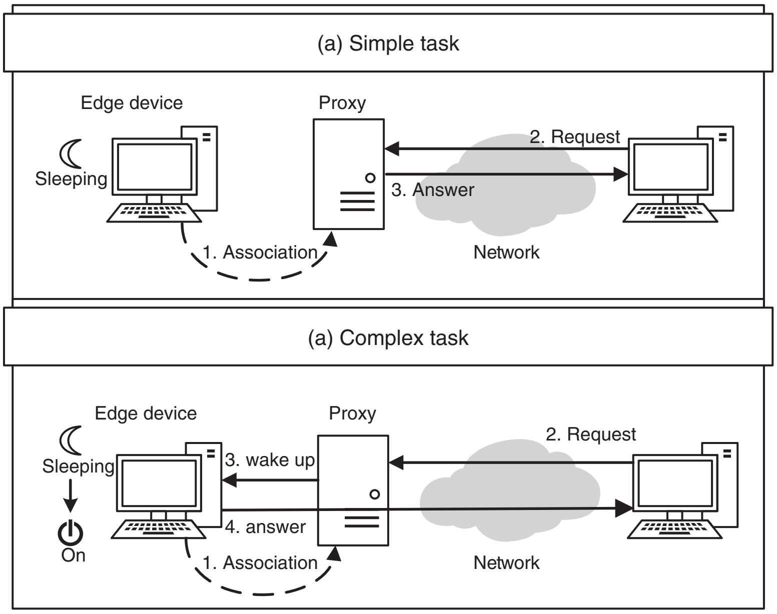

Figure 18.2 Power management proxy operation scheme

Table 18.1 List of proxy-based solutions

Category	Research work
Application-specific proxy	UPnP low power [21]
	SIP catcher [22]
	Proxy for Gnutella [23]
	EE-BitTorrent [24, 25]
Network connectivity proxy	Concept, design and implementation [26, 20, 27–32]
	Somniloquy [33]
	SleepServer [34]
	ECMA-393 [35]

Proxy-based solutions can be further divided into two categories, depending on the kind of proxy they rely upon. We can distinguish between *application-specific proxy* and *network connectivity proxy*. The main proxy-based solutions proposed in the literature are listed in Table 18.1 and are discussed in the following subsections.

18.4.1 Application-Specific Proxy

Proxying is a very common technique in distributed computing. Traditionally (server) proxies have been used in distributed applications to improve the system performance and reduce the network traffic (e.g., *web proxies*). More recently, proxies have been considered in the field of mobile computing to cope with a number of factors, including limited computational capabilities, scarce energy resources, user mobility, and intermittent or weak connectivity. Specifically, a (client) proxy is used as a surrogate of the mobile client on the fixed network, thus allowing the mobile device to be temporarily disconnected from the system, so as to lengthen the lifetime of its battery [18]. In this section, however, we focus on desktop PCs, connected to the power supply with the objective to eliminate energy wastes.

The *UPnP Low Power* architecture [21] represents an example of protocol-specific proxy. Universal Plug and Play (UPnP) [36] is a protocol, defined by the UPnP Forum, that allows devices to seamlessly connect and form spontaneous networks, for example, for data sharing, entertainment, software installation, and so on. The legacy UPnP architecture [36] relies on a distributed discovery protocol that requires all devices to be always powered on in order to respond to discovery messages. The UPnP Low Power architecture defines a low-power proxy to allow devices in the UPnP network to sleep and still be discovered by UPnP control points.

Another example of application-based proxy (in addition to a mechanism for on-demand wake-up) is the *SIP Catcher* [22]. It is a system that allows IP phones to remain in sleep mode for a long time without compromising the application performance. Because of their widespread availability, IP phones are responsible for a significant energy waste, despite their low power consumption. In fact, they remain constantly active but are used for very short periods. SIP is the *Session Initiation Protocol* used to connect IP phones to the Internet. Basically, a user registered with the SIP server and sends an *invite* message to the SIP server when she/he wants to call another IP phone. This locates the recipient IP phone over the Internet and forwards the invite message from the caller. The IP phone responds with a *trying* and a *ringing* message and then rings. At this point, the caller and the called party can start communicating. However, if the IP phone is in sleep mode, it will be obviously unreachable. In this case, a SIP catcher can overcome the drawback. The SIP catcher is a system that runs on the last hop router and acts as a proxy for SIP calls. Specifically, when it detects an *invite* message directed to the sleeping IP phone, it wakes the IP phone up and, once reactivated, transmits the invite message. Meanwhile, the catcher responds to the caller sending the *trying* message. Once reactivated, the IP phone sends the *ringing* message, thus completing the SIP protocol handshake, while being completely transparent to the caller.

Proxying techniques have been also proposed to increase the energy efficiency of *peer-to-peer* (P2P) applications, such as *file sharing/distribution*. Recent studies indicate that a large amount of the overall Internet traffic originates from P2P applications [37]. Nevertheless, P2P file sharing protocols – such as *BitTorrent* and *Gnutella* – have been designed assuming that PCs are always on and, thus, they are not energy efficient. To this end, various solutions have been proposed, including proxy-based architecture (other proposals taking different approaches will be presented in the subsequent sections).

In Ref. [23] the authors present a proxy-based solution for *Gnutella* that also exploits the WoL mechanism. Gnutella uses a flooding mechanism to find files over the overlay network. It defines five different messages, namely *Ping, Pong, Query, Query Hit,* and *Push*. The *Query* and *Query Hit* message are used to find files and respond to query messages, respectively. In the solution presented in Ref. [23] the power management proxy is a microcontroller with limited storage capacity and low energy consumption (much less than the host) and it is positioned on the Ethernet NIC of the host or in a LAN switch. The proxy detects requests for files directed to the sleeping host and wakes it up through a Magic Packet. Thus, the host can serve the requested files. In order to take over for the sleeping host, the proxy shares information with it, about the power state of the host (sleeping or fully powered-on), the IP list of its *neighbors,* and the list of the shared files. The P2P proxy supports only a subset of *Gnutella* functionalities. Specifically, it can start and accept neighbor connections, receive and forward *Query* messages, send *Query Hit* messages, and wake up the sleeping host. Instead, it cannot serve files or store them. When the PC goes to sleep, its TCP connections with neighbors are terminated and established again by the proxy. When the proxy wakes the PC up, the opposite occurs. In both cases, everything happens transparently to the user.

Still in the framework of P2P file sharing, *EE-BitTorrent* [24, 25] is another proxy-based solution for making the *BitTorrent* protocol energy efficient. *BitTorrent* is the most commonly used protocol for P2P file sharing; however it was not designed with energy efficiency in mind. In particular, it requires that a *peer* is always active while downloading a file and remains active, for some time, after completing the download, so as to provide the same file to other peers. *EE-BitTorrent* relies on a *BitTorrent* proxy that serves a large number of *BitTorrent* peers. When a user requests a file, the query is sent to the proxy in a transparent way to the user. The proxy also implements a caching mechanism. Hence, in most cases the requested file is already available on the proxy cache and can be immediately downloaded, thus reducing the download time and energy consumption at the user PC. When the file is not immediately available, the download service is undertaken by the proxy and the user can, thus, turn off her/his PC (Figure 18.3(a)). The proxy gets a copy of the file from the overlay network, acting as a regular *BitTorrent* peer (Figure 18.3(b)). When the copy is available, the proxy wakes the

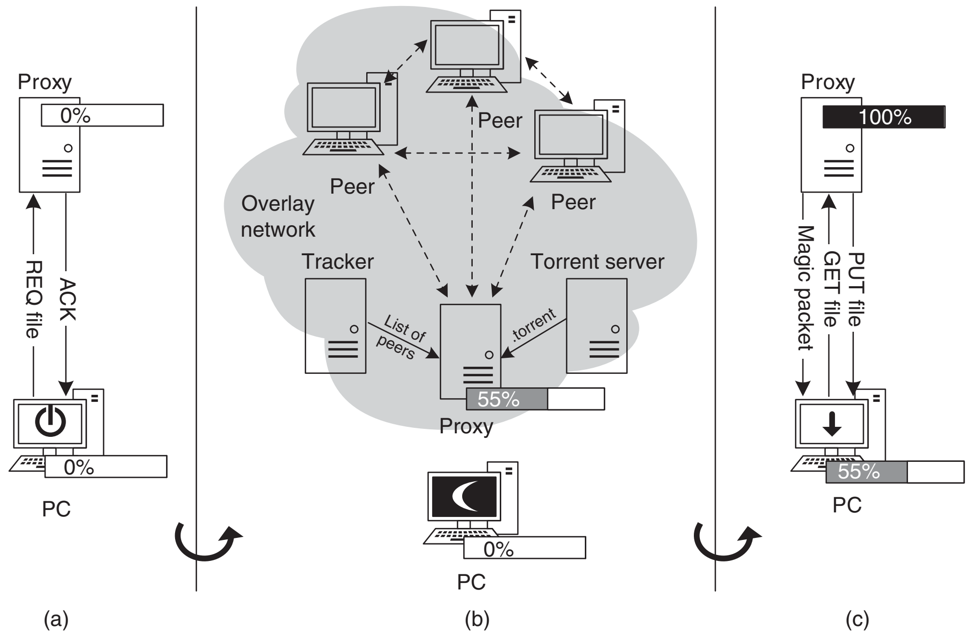

Figure 18.3 EE-BitTorrent operation scheme

PC up, through a Magic Packet, and transfers the file to it (Figure 18.3(c)). If the on-demand wake-up mechanism is not available, the PC can explicitly require the file to the proxy after its reactivation. An important aspect of the system concerns the proxy location. Since the behavior of EE-BitTorrent does not depend on the proxy location, there are many options for placing the proxy, mainly driven by the specific deployment scenario. In an enterprise environment (e.g., a university or business department) the proxy could be located in the same LAN of the served PCs. In a residential scenario, instead, this is not a reasonable option as the proxy would serve only a limited number of PCs (those in the same home). In such a scenario, it could be located in the ISP network and offered as a (free) service to users, or it could be a cloud proxy. Also, a group of users could manage a social (i.e., shared) proxy – connected to the Internet through a high-speed network – for reducing energy consumptions at their home PCs.

Another proxy-based solution, similar to EE-BitTorrent, is presented in Ref. [38], taking into account the efficient sharing (in terms of energy consumption) of files for which only a very limited number of copies are available on the Internet. Downloading such a file often results in a client–server transfer from the peer that provides the file to the peer that requests it, at a very low bit rate. Hence, the benefits of the P2P paradigm are lost, resulting in an increased energy consumption. In the proposed solution, peers are coordinated in such a way that only a limited number of them remains active and act as proxies for the other peers that go in sleep mode.

For the sake of completeness, we also mention here some proxy-based solutions for mobile devices that use a simplified version of the *BitTorrent* protocol [39, 40]. In this case, however, the main objective is to increase as much as possible the battery lifetime of the mobile device.

18.4.2 Network Connectivity Proxy

The proxy-based solutions presented above are *application-specific* as they refer to a particular application or protocol. Therefore, it is necessary to use a different proxy for each specific application. In addition, they typically require the user intervention, that is, they are not transparent. Ideally, a PC should transparently enter sleep mode, whenever it is idle, in order to save energy. At the same time, it should still appear connected and fully operational to the other network devices. This would maximize energy savings while minimizing the impact on the performance of network applications. This goal can be achieved by using a *network connectivity proxy* (NCP), that is, an entity that is capable of maintaining the network presence on behalf of a sleeping PC, managing all packets destined to that PC. The concept of NCP was originally proposed in Ref. [27] for Ethernet networks and, then, extended in subsequent papers [20, 28, 29] for IP networks in general. Key challenges in the design and implementation of an NCP have been addressed in Ref. [30], where the authors propose some possible solutions and show that using an NCP can result in significant energy savings, up to 70%. A sleep proxy similar to the NCP proposed in Ref. [27] is also proposed in Ref. [31].

In order to design an NCP, it is necessary to have a detailed knowledge of the activities carried out by a host (e.g., a PC) to maintain network connectivity. Basically, a host performs a series of actions. Specifically, it replies to periodic ARP requests, generates periodic DHCP requests to maintain the IP address, replies to ICMP messages (e.g., *ping* requests), accepts TPC connections by replying to TCP SYN segments, and, more generally, manages all incoming packets appropriately. A detailed analysis about the packets received by a PC during idle periods, and the related protocols, was carried out in Ref. [29] and, more recently, in Ref. [32]. The latter considers both home and office environments. Once the type and fraction of received packets are known, they can be classified according to the class of actions they require. Hence, NCP requirements can be defined accordingly [26, 32]. In summary, when a packet directed to the sleeping PC is received, the NCP should perform one of the following actions:

(a) Directly respond to the packet.
(b) Discard the packet.
(c) Redirect the packet to another (active) PC for further processing.
(d) Wake up the host and pass the packet for appropriate processing.
(e) Put the packet in a queue to transfer it to the host when this is reactivated.

In addition, the NCP could be instructed to generate periodic requests on behalf of the sleeping PC (e.g., DHCP lease requests for maintaining the IP address) [26].

According to the NCP model outlined above, several practical variants can be envisaged [32]. In fact, the design space is quite large since different solutions may vary in many aspects, including *complexity* (the set of functionalities implemented by the proxy), degree of *transparency* (the possible differences in the user/application behavior with and without the proxy), *deployment* (the place where the proxy is physically located, e.g., individual PC, router/firewall, separate PC), and *implementation* (e.g., device attached to the NIC/motherboard of the PC, external USB-connected device).

In Ref. [32] the authors consider four different proxies with different complexity and compare their performance in terms of energy efficiency. They also propose a general and flexible NCP architecture that can accommodate different design choices and present a simple implementation, wherein the NCP is assumed to be a standalone machine and, thus, it is in charge of maintaining the network connectivity of several PCs in the same LAN. The following NCP variants are considered in the analysis.

- *Proxy-1*: drops all packets classified as *ignorable* and wakes up the PC for handling all other packets.
- *Proxy-2*: drops all packets classified as ignorable, responds directly to protocol packets that require a minimum handling, and wakes up the PC for all other packets.
- *Proxy-3*: performs the same actions as *Proxy-2*, but it is more selective as it wakes up the sleeping PC only when the received packet belongs to a set of user-specified applications.
- *Proxy-4*: performs the same actions as *Proxy-3* with respect to incoming packets. In addition, it can be instructed to wake up the PC to perform scheduled task such as network backups, antivirus updates, software updates, and so on.

The performance comparison is based on traces derived from real measurements, carried out both in home and office environments. The obtained experimental results show that *Proxy-1* is inadequate in office environments and only marginally adequate in home environments. *Proxy-3* provides significant energy savings in both home and office scenarios (it allows the PC to *sleep* for most of the idle time). Instead, the performance of *Proxy-2* largely depends on the specific environment. Specifically, the additional complexity, compared to Proxy-1, makes it a good choice in home environments, although it is not a good candidate for office environments, where the amount of traffic to manage is much higher. Finally, the performance of *Proxy-4* is close to that of *Proxy-3* since scheduled tasks are typically infrequent.

Somniloquy [33] is a private NCP, that is, it is supposed to serve just one PC. It is conceived as an external device, connected to the PC through a USB port, which includes a *low-power* processor capable of running an embedded operating system, flash memory to store data (e.g., files) while the PC is sleeping, and one or more network interfaces to communicate with the external network. When the PC is sleeping, network connectivity is maintained through the NIC of the proxy. The latter uses a packet filter – defined in the form of regular expressions – to select packets and wake up the PC in case of a match. This allows to respond to network applications, such as remote *Secure Shell (SSH)*, file access requests, and VoIP calls, even when the PC is sleeping. In addition, *Somniloquy* can act as an application proxy for some common network applications such as *instant messaging* and *P2P file sharing*. This is accomplished by implementing a lightweight version of the specific application (*stub*) on the proxy. The application stub allows the proxy to manage autonomously the majority of actions required by the application and wakes up the PC only on complex events. A prototype implementation of *Somniloquy* based on the Gumstix[1] platform is described in Ref. [33]. The prototype includes a 200 MHz XScale processor with 2 GB of flash memory and 64 MB of RAM, a wired Ethernet (or wireless WiFi) NIC for connectivity, and two USB ports (one for sleeping/waking up the PC, the other one for relaying data received from NIC to the PC), and runs a version of Embedded Linux that supports a full TCP/IP protocol stack. To give an idea of the amount of energy saved by *Somniloquy* we can consider that a common PC consumes approximately 100 W in normal operating conditions, while the total power consumed by the PC in sleep mode and the external device implementing *Somniloquy* is approximately 5 W.

Somniloquy and the different proxy variants considered in Ref. [32] are not able to preserve TCP connections when the PC is in sleep state. Instead, this issue is specifically addressed in

[1] http://www.gumstix.com/

Ref. [26]. In addition to the outlined tasks (i.e., discarding ignorable packets, directly replying to packets that require minimal actions, and waking up the host when needed), the NCP proposed in Ref. [26] is also able to maintain TCP connections and UDP data flows. This is achieved by splitting the TCP connection at the proxy (see Section 18.6.1 for details about *splitting*). We assume that the proxy runs in the same network of the PC (e.g., on a router) and can, thus, cover several hosts. TCP packets destined to a given PC are buffered locally at the NCP when the PC is sleeping, and are later relayed, when the PC is awake again. Queuing of packets for later processing may actually make sense for some network applications such as *instant messaging* and *SSH*.

SleepServer [34] is another *proxy-based* solution that allows a host (e.g., PC, printer) to go in sleep mode while remaining reachable at the application layer. It does not require any change to the network infrastructure or any additional hardware, but only software agents installed on hosts. Indeed, it is completely implemented in software, using virtualization techniques. Specifically, the proposed architecture is physically composed of one or more SleepServer (SSR) machines on the same subnet as the hosts (but it can also work for hosts on different subnets using the VLANs mechanism). Each SSR serves a set of hosts and contains their images, in the form of virtual machines (VM). An SSR maintains the presence of the hosts in the network when they are in a state of sleep. On each SSR, a *SSR-Controller* is installed. It is a software component that manages (i) the creation of host images, (ii) the communication between hosts and their images, (iii) the resources allocation, and (iv) sharing among the images, providing isolation between images. Each host has also a software component, the *SSR-Client*, that connects the host to the SleepServer, passing its MAC and IP addresses and its firewall configurations. Before going to sleeping state, it sends its applications' state and all its open TCP and UDP ports to the SSR-Controller that creates the host image on the SSR machine. The image uses the same configuration parameters as the corresponding host. Hence, while the host is asleep, the image interacts with the network on behalf of it. If the host interaction is required, the SSR-Controller wakes up the host and disables its image on the SleepServer. In Ref. [34] it is shown that this solution allows significant energy savings (about 60–80%) and is able to support heterogeneous operating systems. Due to its high scalability, it is especially suitable for enterprise LAN environments with a large number of hosts (PCs, printers, etc.) connected to the network.

Finally, ECMA-393 [35] is a standard, adopted in February 2010, that specifies the maintenance of network connectivity and presence by proxies in order to extend the sleep duration of hosts, so as to save energy. The standard defines the behavior and the architecture of the proxy. In particular, it specifies the capabilities that a proxy may expose to a host, the information that must be exchanged between a host and a proxy, the proxy behavior with 802.3 (Ethernet) and 802.11 (WiFi) NICs and, more generally, the behavior of a proxy, including responding to packets, generating packets, ignoring packets, and waking up the host.

18.5 Context-Aware Power Management

In order to minimize energy consumption, PCs and other edge devices should be ideally turned off, or put in sleep mode, whenever they are not used. Also, they should be switched on again as soon as the user needs to use them. However, manually managing the power state of edge devices could be too onerous and frustrating for users. For these reasons, *Context-Aware Power Management* (*CAPM*) strategies have been developed, that is, strategies that use context information to automatically manage the power state of a device at runtime. Essentially, they aim to determine – by means of proper sensors – if the user *is using*, *is not using,* or *is about to use* a device. Through this information, the system is able to change the power state of the device, thus resulting in energy savings, while providing, at the same time, an acceptable service quality to the user.

CAPM strategies can also be used to optimize the energy consumption of specific components of a PC – such as hard drive, NIC, CPU, or display – according to the actual usage by the user. For example, the authors of Ref. [41] propose a solution that relies on a camera to determine if a user is looking at the display and turns off the display if the user is not present. Specifically, the CAPM system periodically acquires images by the laptop camera, which are processed by a *face detector* algorithm. If the user's face is not found, the system turns off the display.

However, more significant energy savings can be achieved by switching the entire machine to a low-power state during idle periods. Obviously, the power management system cannot turn off the PC as soon as the interaction with the user ceases, but it must infer from the context whether the user has actually stopped using the computer. In fact, transitions from one state to another have a significant cost in terms of

(i) *energy*: switching on a sleeping PC causes a considerable power consumption (much higher than the consumption in idle state);
(ii) *time*: in order to resume a PC to a fully operational state, tens of seconds could pass and, during this time interval, the user is unable to use her/his PC;
(iii) *lifetime of the device*: switching off and on frequently a device may reduce its lifetime.

The remarks above suggest that a too aggressive power management strategy, characterized by frequent shutdowns, would not lead to significant energy savings (due to the consumption during the wake-up phase) and could be highly frustrating for the user (forced to wait long resume periods). Therefore, it is possible to define a *break-even* time, that is, the minimum time interval that a device must spend in a sleep state in order to justify the passage to the low-power state and the subsequent reverse transition. For PCs, this time is in the order of minutes. Hence, a CAPM strategy requires accurate information to predict whether an idle period is long enough to justify the cost of the state change.

The required information can be obtained by means of a *location-aware* system, that is, a system able to determine the user's position with respect to the PC. Depending on the distance from the machine, the system can, thus, evaluate if the user is leaving her/his workplace or has temporarily discontinued the use of the PC. Hence, it can decide whether or not to put the PC into standby mode. Similarly, it can recognize when the user is back to the PC and switch it on, if necessary. In the literature there are various power management approaches that rely on the user's location estimation. They can be classified into two main categories that are discussed below.

The first category includes CAPM mechanisms that rely on very accurate information about the user's location, obtained through sophisticated location systems. For instance, the solution presented in Ref. [42] exploits an ultrasonic system that provides the user's location with high accuracy. Alternatively, a ultra-wideband (UWB) radio system can offer a good compromise between accuracy (below 1 m) and deployment costs. This kind of approach can be defined as *reactive*, since the system defines some *spatial zones* and triggers special events (i.e., switch on or suspend a PC) when a user enters or leaves a specific zone. Obviously, although these solutions provide excellent performance, their complexity and costs are typically very high.

The second category includes solutions that do not rely on very accurate location information. They typically exploit low-power sensors that provide only approximate information about the user's position. For instance, they check the radio connectivity of personal mobile devices, such as smartphones, to infer the presence/absence of the user in the working area [43, 44]. The solution proposed in Ref. [44] uses a policy called *Sleep/Wake-up on Bluetooth*. Specifically, the PC periodically runs the *Bluetooth discovery* procedure to detect the presence of the user's Bluetooth phone. If the latter is not discovered, the PC enters the standby mode. Then, when the user comes back within the Bluetooth coverage area, a nearby server detects the phone and wakes up her/his PC. Approaches falling in this second category can be defined *proactive*, as the system guesses the user's intentions and changes the power state of the PC accordingly. They are easy to implement; however, they are much less accurate than the previous ones. Hence, they may cause undesired shutdowns or activations of the PC.

When using low-power low-accuracy sensors, data provided by different sensors can be combined together, using AI techniques, in order to get more accurate location information. For instance, in Ref. [45] the authors exploit not only the information provided by Bluetooth phones (as in Ref. [44]), but also other context information provided by acoustic sensors and software sensors that monitor the user's activity on the PC. All these data are then processed using *Bayesian inference techniques*, so as to infer the user's position and activity, and act on the power state of the PC accordingly. Another interesting solution, called *non-intrusive location-aware power management scheme* (*NAPS*) is presented in Ref. [46]. NAPs is specifically designed for PCs and does not require very accurate location information. It divides the space around a PC in some *virtual zones* and then uses the *received signal strength indicator* (RSSI) of a sensor node, carried on by the user, to estimate the zone where the user is located. According to the estimated distance from the computer, the system performs different actions. If the user moves away from her/his PC, the system acts first at the application level, closing unnecessary applications that use the CPU (e.g., web browsers) and, then at the device level, switching off some components, such as video and hard disk. If the user moves further away, the PC is put into the sleep state.

Obviously, the efficiency of a CAPM solution depends on several factors. For instance, the use of very accurate location methods, or the addition of context information, can increase the performance of the power management system. On the other hand, we also need to consider the additional cost of sensors and their energy consumption. The study in Ref. [47] analyzes the costs and potentialities of CAPM systems. The obtained results show that context information and location methods must be carefully chosen in order to maximize the ratio between energy saved through power management and energy consumed by sensors. At the same time, the system should be capable not only of saving energy, but also be transparent to the user. Forcing the user to manually turn on her/his PC, or to wait long resuming times, could be irritating and lead the user to disable the power management system. It is thus preferable to reduce the intrusiveness of the system, at the cost of a slightly higher energy usage. Anyway, as shown in Refs. [42, 45, 47], CAPM systems are able to produce significant energy savings, although occasional unnecessary shutdowns or switches-on are unavoidable. Obviously, CAPM systems can never achieve an optimal performance, because their effectiveness strictly depends on the particular use that the user makes of her/his PC and requires an accurate prediction of the user's future intentions.

18.6 Power-aware Protocols and Applications

In this section we describe some techniques for designing *power-aware* protocols and applications. All common protocols (e.g., TCP) have been implicitly designed assuming that the hosts on which they run are continuously active and, therefore, they are not power-aware. To be used in hosts that can switch between sleep and active modes, these protocols (applications) must be properly adapted. To this end, there are two viable approaches: (i) modifying existing protocols (applications) in order to make them power-aware, or (ii) defining new protocols and applications from scratch.

To implement new energy-efficient protocols and applications, it is recommended to use specific software tools, such as the ones presented in Refs. [48, 49]. These tools perform an energy profiling of the protocol/application in all its parts and assist the developer during programming, so that she/he can make design choices aimed at reducing energy consumption. However, the approach (i) is the most widely used for very common protocols and applications. Therefore, in the following we refer to techniques to modify existing protocols and applications, in an energy-efficient perspective. According to Ref. [49], it is possible to act both at the transport and the application layer. The two approaches are discussed below.

18.6.1 Transport Protocols

In this section we survey solutions working at the transport layer of the networking protocol stack. The main advantage of this approach is that energy-aware capabilities added to the transport protocol (e.g., TCP) can be exploited by all the applications running on top of it.

With reference to TCP, many research activities have been carried out to make the protocol energy efficient in the context of mobile/wireless networking [50–54], where an improvement in the efficiency of the networking subsystem can significantly increase the lifetime of the mobile computer. Although the above-mentioned solutions could also apply to stationary PCs, in the following we focus mainly on solutions specifically targeted at energy efficiency in stationary/wired environments.

In Ref. [55] the authors analyze the energy cost of TCP, due to computational activities, by investigating the protocol consumption on different hardware platforms and with different (Unix and Linux) operating systems. In addition, they propose solutions to improve the energy efficiency of the existing TCP implementations.

The computational energy cost of TCP includes the energy consumption due to the following activities:

1. moving data from the user space into kernel space (*user-to-kernel copy*), as data sent through a TCP socket is first queued in the socket buffer and then copied into kernel space for further processing;
2. copying packets to the network card (*kernel-to-NIC copy*);
3. processing in the TCP/IP protocol stack, including the cost for computing the checksum, preparing ACKs, responding to time-out events and to triple duplicate ACKs, and other costs (window maintenance, estimation of Round Trip Time, interrupt handling, etc.).

The experimental results clearly show that a large part of the energy consumption associated with TCP is because of the copy operations, while only about 15% of the consumed energy is linked to TCP processing. Considering a more detailed breakdown of the energy costs, the authors estimate the consumption for each phase of the protocol. In particular, the step of computing the checksum alone covers about 30% of the energy consumption of the entire TCP processing. In order to reduce the energy consumption, the authors present some solutions. For instance, using *zero copy* technique – that is, directly copying the user data from the user buffers to the NIC – it is possible to skip the user-to-kernel copy phase. Instead, in order to reduce the kernel-to-NIC copy cost, two methods are proposed, aiming at decreasing the number of copies: (i) maintain the TCP send buffer on the NIC itself and (ii) maximize the data transfer size from the kernel to the NIC. Using all of these mechanisms, the authors showed that it is possible to achieve a reduction in the overall cost at a sender of about 30%.

A significant problem concerning energy efficiency in TCP is the maintenance of a connection when a PC goes to sleep. Many applications (e.g., *SSH*, *Instant Messaging*) need a permanent TCP connection between the client and the server. To maintain the persistence of the connection, hosts must generate (and respond to) periodic *keep-alive messages* even when the TCP connection is idle, that is, when neither the client nor the server needs to send data. These messages can be generated directly by the TCP protocol (at least once every 2 hours) or by the application. Nevertheless, when a PC is in the sleep state, its processor is stopped and, thus, it cannot process incoming packets and reply with ACK packets. After a predetermined time-out period, data packets (e.g., keep-alive messages) are assumed to get lost and retransmitted. If no answer is received after a certain number of attempts, the sending host drops the connection, cleaning up all the resources associated with the connection. The application is then notified that the connection is closed, resulting in an error, since the application expected a persistent connection. In the literature, two solutions have been proposed to modify the TCP protocol in order to overcome this problem, namely, *Green TCP/IP* [56] and *splitting* [29].

Green TCP/IP, proposed in Ref. [56], adds the concept of *connection sleep state* to the legacy TCP protocol. Basically, the Green TCP/IP client notifies the Green TCP/IP server that it is going to sleep. Thus, the server logically keeps the connection alive but does not send any data or ACK packet to the sleeping client. Besides, the *socket* associated with the connection is blocked in the server so as to avoid excessive queuing of data to send. When the client wakes up, it has to notify the server – simply sending a data packet on the sleeping connection – and, thus, the data flows between the server and client can be immediately resumed. Obviously, the Green TCP/IP must be compatible with legacy TCP, in order to allow the coexistence with regular TCP/IP hosts. To this end, a new *TCP_SLEEP* option has been introduced into the header of the segment. This field is used to inform the server that the client is entering the sleep mode. When the server receives a packet containing the *TCP_SLEEP* option, it will avoid dropping the connection for that client. Considering that – according to the TCP protocol – a host ignores each option that it does not understand, this solution is, therefore, backward compatible.

An alternative approach to Green TCP/IP is proposed in Ref. [29] and is based on *splitting*. The main goal of this solution is to allow an application to receive replies for keep-alive messages even if the host at the other end of the TCP connection is sleeping. In addition, it provides a quick way to fully resume the TCP connection toward a sleeping host in case the application needs data to be transmitted. In order to realize this solution, the TCP connection at each host can be split in two parts, adding a *shim layer* between the *socket interface* and the application. This shim layer presents a socket interface to the application (so the application does not need any changes) and uses the existing socket layer of the TCP software implementation. The shim layer 'deceives' applications making them see an established connection at

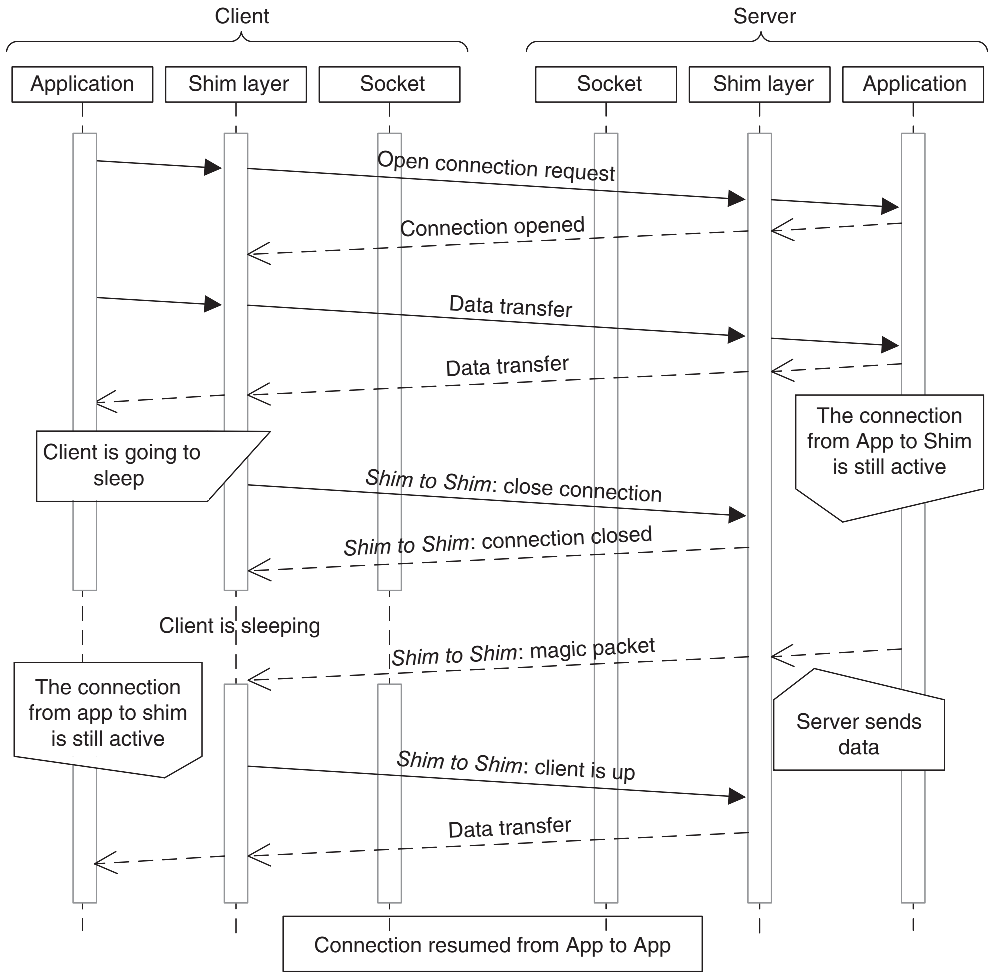
Client
Server
Application
Shim layer
Socket
Socket
Shim layer
Application
Open connection request
Connection opened
Data transfer
Data transfer
The connection from App to Shim is still active
Client is going to sleep
Shim to Shim: close connection
Shim to Shim: connection closed
Client is sleeping
Shim to Shim: magic packet
The connection from app to shim is still active
Server sends data
Shim to Shim: client is up
Data transfer
Connection resumed from App to App

Figure 18.4 TCP connection splitting mechanism

all times. Actually, splitting the TCP connection, applications continue to see the connection with the shim layer, even when the TCP connection between the client and the server hosts has been closed. The shim layer's functionality is used only when power management is enabled (i.e., the client goes into sleep mode) and it works transparently to the application. Figure 18.4 shows how the splitting mechanism works. When the client enters the sleep state, its shim layer notifies the corresponding layer in the server, so as to drop the current TCP connection. If the server wants to send data to a sleeping client, its shim layer preliminarily wakes up the client, through a wake-up message. When the client is up again, the shim layer reestablishes the TCP connection with the opposite shim layer so that data transfer can start.

18.6.2 Application-Layer Protocols

In addition to improving transport-layer protocols at kernel level, it is also possible to design energy-efficient applications at user level. In this section we review the main energy-aware application-layer protocols proposed in the literature.

Green Telnet [57] is a very relevant example of power-aware application-layer protocol, and the approach used in *Green Telnet* can be easily extended to other client/server applications. The goal is to allow clients to go into sleep state without losing their session with the server. For this purpose, the authors propose some changes both at server and client side. Basically, *Green Telnet* decouples the state of the TCP connection from the state of the Telnet server, by an abstraction process. The application server (*gtelnetd*) does not operate directly on a socket but uses an intermediate buffer. In practice, when a client goes to the sleep state, it notifies the server – with a special message – of its intention to interrupt the communication. While the client is sleeping, the application at the server side continues to write data. Nevertheless, such data is not sent immediately to the client but is stored in the buffer. When the client wakes up, it creates a new TCP connection with the server (on a proper port communicated by the server before the power state change) and receives all the data stored during the inactive period. The software has been implemented through three processes at the server side that control the phase of client reconnection, the sending and the receiving of data to and from the buffer shared with the *gtelnetd demon,* respectively. Since *gtelnetd* acts on the intermediate buffer, all the sleep/wake-up operations occur transparently. The original Telnet protocol has been modified to implement Green Telnet by inserting a new field in the packet header and defining the new control messages that regulate the communication of the power state change.

Various methods for reducing the energy consumption have also been proposed for P2P protocols. A detailed survey of energy-efficient P2P systems and applications is available in Ref. [58]. We describe below the main solutions proposed in the literature, from a protocol perspective.

Significant energy savings can be achieved in P2P systems by properly allocating tasks to peers. A “client” peer that needs a service has to find a “server” peer that can satisfy its request. Hence, this “server” peer can be suitably chosen so as to minimize the energy consumption. In Refs. [59] and [60] the authors consider a Web application on P2P overlay networks. This is a typical example of a transaction-based application, in which the main cost for the fulfillment of the request consists in the consumption of CPU resources (i.e., in processing) while the expense for the content distribution is absolutely marginal. Thus, the authors present a computation model and a power consumption model in order to describe how processes run on a server peer and how much energy their execution consumes. In particular, two versions – a simple and a multilevel version – of the power consumption model are proposed. The simple model is more suitable for PCs with one CPU, whereas a server computer with multiple CPUs follows the multilevel model. Exploiting these models, the authors propose an algorithm by which a client peer can select a server peer in a set, so as to satisfy some constraints (e.g., temporal constraints) and reduce power consumption. Simulation results show energy savings up to 12.2%, compared to traditional Round Robin algorithms.

The most common use of peer-to-peer applications consists in file sharing. In this context, *BitTorrent* is the most commonly used protocol over the Internet. Hence, a number of solutions has been proposed to make it energy efficient. *Green BitTorrent* [61] is a modified version of the

BitTorrent protocol that allows *peers* that have completed their download process (*seeds*) and that are not currently involved in any upload operation, to go into sleep mode. From the point of view of a generic peer *P*, the other peers in the same *swarm* can be in one of the following states: *connected*, if the TCP connection is active; *sleeping*, if the peer is disconnected but the TCP connection could be reestablished and *unknown*. When *P* detects a peer disconnection, it sets, in a list, the state of that disconnected host to "sleeping." However, the TCP connection is not dropped. When the number of connected peers is less than a predefined threshold, peer *P* can explicitly wake up a sleeping peer, by sending a special wake-up message (i.e., a Magic Packet or another WoL mechanism). Once *P* completes its download, it starts an *inactivity timer* to clock the idle periods (i.e., without upload operations). After the timer expiration, the peer can go to sleep. Basically, it communicates its intentions sending a message to all the connected peers and, then, it enters the sleep state. While *P* is in this state, it can receive a wake-up message from another peer. In this case, it establishes a TCP connection with the peer that sent the message and starts exchanging data with it. Green BitTorrent is compatible with the legacy version of the protocol, even if peers using legacy BitTorrent protocol experience a slight performance degradation, in terms of higher average download time. It allows to obtain a considerable reduction of the energy consumption [61].

18.7 Conclusions

In this chapter we have addressed the problem of energy-efficient protocol design for reducing energy wastes because of edge devices (i.e., PC, printers, IP phones, etc.) that are typically left on even when they are not needed. Specifically, we have surveyed the main solutions proposed in the literature to address this problem and presented a taxonomy. According to the proposed taxonomy, we can classify these solutions into four main categories, namely, *on-demand wake-up, proxying, context-aware power management,* and *power-aware protocols*.

Solutions exploiting on-demand wake-up allow to turn on a host remotely, by sending a special packet called Magic Packet. These solutions are very common and used in many power management systems, also for large-scale distributed systems. However, they exhibit a number of limitations, mainly in terms of security and privacy. Some proprietary solutions have been proposed to overcome these issues. However, they have not yet reached the same degree of diffusion and compatibility as the original solution.

Solutions based on proxying allow a host to delegate the answer to certain types of requests to a proxy and go to sleep. Both application-specific and network connectivity proxies are available. Network connectivity proxies are application independent and allow significant energy savings, up to 70%, depending on the usage model of the host, while guaranteeing continuous network connectivity. A network connectivity proxy can be either a *private* proxy serving just one host or a *shared* proxy capable of serving many hosts. The former solution is suitable for PCs, while the latter one is more appealing for large distributed systems.

Context-aware power management strategies allow to manage the power state of a host by exploiting proper context information. They rely on specific sensors to determine the user's position and turn off the PC when the user is far from it. Several approaches can be used to obtain the required context information. They can make use of accurate and expensive sensors or low-cost general devices (such as Bluetooth phones). Also, it is necessary to consider the additional energy consumed (e.g., by sensors), and the intrusiveness of the power management system. Ideally, power management should be totally transparent to the user. Instead, forcing the user to manually turn on her/his host or to wait long resuming times due to wrong decisions could be irritating and lead the user to disable power management. Currently, to the best of our knowledge, no such strategy is used in commercial systems. This is mainly because it is very difficult to predict the users' intentions. However, context-aware power management is a stimulating research field.

Finally, the last approach consists in designing power-aware protocols and applications. In particular, we have focused on methods to modify existing protocols and applications in order to make them energy efficient. While designing new energy-efficient solutions from scratch would be more effective, some protocols (e.g., TCP, BitTorrent) are so common that it is almost impossible to reimplement them from scratch. Hence, modifying existing protocols and applications in a green perspective is the only way to achieve energy efficiency while preserving backward compatibility.

Before concluding this chapter, it may be worthwhile emphasizing that the previous approaches are not necessarily alternative. Instead, some of them can coexist. For instance, network connectivity proxies typically rely on magic packets to wake up the served host, when necessary. Also, power-aware protocols and applications can coexist with context-aware power management strategies or proxy-based solutions.

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 U.S.: detailed report and appendices," Technical Report LBNL-45917, Energy Analysis Department, Lawrence Berkeley National Laboratory, 2001.

[2] B. Raghavan, J. Ma, "The Energy and emergy of the Internet", Proceedings of ACM Workshop on Hot Topics in Networks (Hotnets 2011), Cambridge, Massachusetts, USA, November 14–15, 2011.

[3] P. Bertoldi, B. Hirl, N. Lab, "Energy efficiency status report 2012 – electricity consumption and efficiency trends in the EU-27", JRC Scientific and Policy Reports, European Commission, Joint Research Centre, Institute for Energy and Transport, 2012.

[4] S. Ruth, "Green IT – more than a three percent solution", IEEE Internet Comput. Mag., vol. 13, no. 4, pp. 74–78 2009.

[5] R. Bolla, R. Bruschi, F. Davoli, F. Cucchietti, "Energy efficiency in the future Internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures", IEEE Commun. Surveys Tutorials, vol. 13, no. 2, 2011.

[6] R. Bolla, R. Bruschi, K. Christensen, F. Cucchietti, F. Davoli, S. Singh, "The potential impact of green technologies in next generation wireline networks – is there room for energy savings optimization?", IEEE Commun. Mag., vol. 49, no. 8, pp. 80–86, 2011.

[7] S. Karayi, "The PC energy report, 1E", National Energy Foundation (NEF), London [Online], 2007. Available at: http://www.1e.com/energycampaign/downloads/1E_reportFINAL.pdf

[8] G. Newsham, D. Tiller, "A case study of the energy consumption of desktop computers", Proceedings of IEEE Industry Applications Society Annual Conference, Houston, Texas, USA, October 4–9, 1992.

[9] K. Christensen, "The next frontier for communications networks: power management", Proceedings of SPIE - Performance and Control of Next-Generation Communications Networks, Vol. 5244, pp. 1–4, 2003.

[10] K. Christensen, P. Gunaratne, B. Nordman, A. George, "The next frontier for communications networks: power management", Comput. Commun., vol. 27, no. 18, pp. 1758–1770, 2004.

[11] Magic Packet Technology, White Paper, Publication# 20213, Rev: A, Amendment/0, 1995.

[12] "Remote Wake-Up: Intel® Network Adapters User Guide", Intel Corporation [Online], 2008, Available at: http://driveragent.com/archive/17228/image/7-0-93.

[13] L. Chiaraviglio, M. Mellia, "Polisave: efficient power management of campus PCs", Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2010), Split, Dubrovnik, Croatia, September 23–25, 2010.

[14] “Advanced Configuration and Power Interface Specification, revision 5.0”, Hewlett-Packard, Intel Corporation, Microsoft, Phoenix Technologies, Toshiba, December 6 [Online], 2011. Available at: http://acpi.info /DOWNLOADS/ACPIspec50.pdf.

[15] K. Kurowski, A. Oleksiak, M. Witkowski, “Distributed power management and control system for sustainable computing environments”, Proceedings of International Conference on Green Computing (IGCC 2010), Chicago, Illinois, USA, August 15–18, 2010.

[16] “Hardening Measures Built into Intel® Active Management Technology”, Intel ®, 2010 [Online]. Available at: http://software.intel.com/en-us/articles/hardening-measures-built-into-intel-active-management-technology/.

[17] “Intel® Core™ vPro™ Technology: Intelligence Adapts to Your Needs”, White Paper, Intel® [Online]. Available at: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/remote-support-vpro-intelligence-that-adapts-to-your-needs-paper.pdf

[18] E. Pitoura, G. Samaras, “Data Management for Mobile Computing”, Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1997.

[19] G. Anastasi, M. Conti, E. Gregori, A. Passarella, “Performance comparison of power saving strategies for mobile web access”, Perform. Eval., vol. 53, no. 3–4, pp. 273–294, 2003.

[20] B. Nordman, K. Christensen, “Improving the energy efficiency of ethernet-connected devices: a proposal for proxying”, White Paper, Version 1.0, Ethernet Alliance, 2007.

[21] UPnP Low Power Architecture V1.0, UPnP Forum, August 27, 2007 [Online]. Available at: http://www.upnp .org/specs/lp.asp.

[22] M. Jimeno, “The SIP Catcher: a Service to Enable IP Phones to Sleep” [Online]. Available at: http://www .youtube.com/watch?v=KdAm4olcVoo.

[23] M. Jimeno, K. Christensen, “A prototype power management proxy for Gnutella peer-to-peer Ffile sharing” Proceedings of IEEE Conference on Local Computer Networks (LCN 2007), Dublin, Ireland, October 15–18, 2007.

[24] G. Anastasi, M. Conti, I. Giannetti, A. Passarella, “Design and evaluation of a BitTorrent proxy for energy saving”, Proceedings of IEEE Symposium on Computers and Communications (ISCC 2009), Sousse, Tunisia, July 5–8, 2009.

[25] G. Anastasi, I. Giannetti, A. Passarella, “A BitTorrent proxy for green Internet file sharing: design and experimental evaluation”, Comput. Commun., vol. 33, no. 7, pp. 794–802, 2010.

[26] M. Jimeno, K. Christensen, B. Nordman, “A network connection proxy to enable hosts to sleep and save energy”, Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC 2008), Austin, Texas, USA, December 7–9, 2008.

[27] K. Christensen, F. Gulledge, “Enabling power management for network-attached computers”, Int. J. Netw. Manag., vol. 8, no. 2, pp. 120–130, 1998.

[28] K. Christensen, B. Nordman, R. Brown, “Power management in networked devices” IEEE Comput., vol. 37, no. 8, pp. 91–93, 2004.

[29] C. Gunaratne, K. Christensen, B. Nordman, “Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed”, Int. J. Netw. Manag., vol. 15, no. 5, pp. 297–310, 2005.

[30] R. Khan, R. Bolla, M. Repetto, R. Bruschi, M. Giribaldi, “Smart proxying for reducing network energy consumption”, Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2012), Genoa, Italy, July 8–11, 2012.

[31] S. Cheshire, “Method and apparatus for implementing a sleep proxy for services on a network”, United States Patent N. 7,330,986, February 12, 2008.

[32] S. Nedevschi, J. Chandrashekar, B. Nordman, S. Ratnasamy, N. Taft, “Skilled in the art of being idle: reducing energy waste in networked systems”, Proceedings of USENIX Symposium on Networked System Design and Implementation (NSDI, 2009), Boston, Massachusetts, USA, April 22–24, 2009.

[33] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, R. Gupta, “Somniloquy: augmenting network interfaces to reduce PC energy usage”, Proceedings of USENIX Symposium on Networked System Design and Implementation (NSDI, 2009), Boston, Massachusetts, USA, April 22–24, 2009.

[34] Y. Agarwal, S. Savage, and R. Gupta, “SleepServer: energy savings for enterprise PCs by allowing them to sleep”, Proceedings of USENIX Annual Technical Conference, Boston, Massachusetts, USA, 2010.

[35] Standard ECMA-393 “ProxZzzy for Sleeping Hosts, 1st edition,” 2010.

[36] UPnP Device Architecture, UPnP Forum [Online]. Available at: http://www.upnp.org/standardizeddcps/ default.asp.

[37] H. Schulze, K. Mochalski, IPOQUE – Internet Study 2008/2009, Leipzig, Germany, 2007.

[38] H. Hlavacs, R. Weidlich, T. Treutner, “Energy efficient peer-to-peer file sharing”, J. Supercomput., vol. 62, no. 3, pp. 1167–1188, 2012.

[39] I. Kelenyi, A. Ludanyi, J. Nurminen, “BitTorrent on mobile phones-energy efficiency of a distributed proxy solution”, Proceedings of International Green Computing Conference (IGCC 2010), Chicago, Illinois, USA, August 15–18, 2010.

[40] I. Kelenyi, A. Ludanyi, J. Nurminen, I. Pusstinen, “Energy-efficient mobile BitTorrent with broadband router hosted proxies”, Proceedings of IFIP Wireless and Mobile Networking Conference (WMNC 2010), Budapest, Hungary, October 13–15, 2010.

[41] A. Dalton, C. Ellis, “Sensing user intention and context for energy management”, Proceedings of Workshop on Hot Topics in Operating Systems (HotOS IX), Lihue, Hawaii, USA, May 18–21, 2003.

[42] R. K. Harle, A. Hopper, “The potential for location-aware power management”, Proceedings of International Conference on Ubiquitous Computing (UbiComp 2008), Seoul, South Korea, September 21–24, 2008.

[43] M. Youssef, A. Agrawala, “The Horus WLAN location determination system”, Proceedings of International conference on Mobile Systems, applications, and services (MobiSys 2005), Seattle, Washington, USA, 6–8 2005.

[44] C. Harris, V. Cahill, “Power management for stationary machines in a pervasive computing environment”, Proceedings of Hawaii International Conference on System Sciences (HICSS 2005), Hawaii, USA, January 3–6, 2005.

[45] C. Harris, V. Cahill, “Exploiting user behaviour for context-aware power management”, Proceedings of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob 2005), Montreal, Canada, August 22–24, 2005.

[46] Z.-Y. Jin and R. K. Gupta, “RSSI based location-aware PC power management”, In Workshop on Power Aware Computing and Systems (HotPower 2009), Big Sky, Montana, USA, October 10, 2009.

[47] C. Harris, V. Cahill, “An empirical study of the potential for context-aware power management”, Proceedings of International Conference on Ubiquitous computing (UbiComp 2007), Innsbruck, Austria, September 16–19, 2007.

[48] A. Kansal, F. Zhao, “Fine-grained energy profiling for power-aware application design”, ACM SIGMETRICS Perform. Eval. Rev., vol. 36, no. 2, pp. 26–31, 2008.

[49] W. Baek, T. Chilimbi, “Green: a framework for supporting energy-conscious programming using controlled approximation”, Proceedings of ACM SIGPLAN conference on Programming language design and implementation (PLDI 2010), Toronto, Canada, June 5–10, 2010.

[50] A. Ayadi, P. Maille, D. Ros, “TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption”, Proceedings of IFIP International Conference on New Technologies, Mobility and Security (NTMS 2011), Paris, France, February 7–10, 2011.

[51] C. Song, S. W. Turner, H. Sharif, “An energy-efficient TCP quick timeout scheme for wireless LANs”, Proceedings of IEEE International Performance, Computing, and Communications Conference (IPCCC 2003), Phoenix, Arizona, USA, April 9–11, 2003.

[52] F. Keceli, I. Inan, E. Ayanoglu, “Fair and efficient TCP access in IEEE 802.11 WLANs”, Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2008), Las Vegas, Nevada, USA, March 31–April 3, 2008.

[53] N. Cho, K. Chung, “TCP-New veno: the energy efficient congestion control in mobile ad-hoc networks”, Proceedings of International Conference on Embedded and Ubiquitous Computing (EUC 2006), Seoul, South Korea, August 1–4, 2006.

[54] A. Seddik-Ghaleb, Y. Ghamri-Doudane, S. Senouci, “A performance study of TCP variants in terms of energy consumption and average goodput within a static ad hoc environment”, Proceedings of International Conference on Wireless Communications and Mobile Computing (IWCMC 2006), Vancouver, Canada, July 3–6, 2006.

[55] B. Wang, S. Singh, “Computational energy cost of TCP”, Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), Hong Kong, China, March 7–11, 2004.

[56] L. Irish, K. Christensen, “A’Green TCP/IP’ to reduce electricity consumed by computers”, Proceedings of IEEE Southeastcon 1998, Orlando, Florida, USA, April 24–26, 1998.

[57] J. Blackburn, K. Christensen, “Green telnet: modifying a client-server application to save energy”, Dr. Dobb’s J., vol. 414, pp. 33–38, 2008.

[58] A. Malatras, F. Peng, B. Hirsbrunner, “Energy-efficient peer-to-peer networking and overlays”, Chapter 20 in Handbook of Green Information and Communication System, M. S. Obaidat, A. Anpalagan, I. Woungang, Eds, Elsevier: Academic Press, 2012.

[59] T. Enokido, A. Aikebaier, M. Takizawa, “A model for reducing power consumption in peer-to-peer systems”, IEEE Syst. J., vol. 4, no. 2, pp. 221–229, 2010.
[60] T. Enokido, A. Aikebaier, M. Takizawa, “Process allocation algorithms for saving power consumption in peer-to-peer systems”, IEEE Trans. Ind. Electron., vol. 58, no. 6, pp. 2097–2105, 2011.
[61] J. Blackburn, K. Christensen, “A simulation study of a new green BitTorrent”, Proceedings of International Workshop on Green Communications (GreenComm 2009), Dresden, Germany, June 18, 2009.