Smart Proxying: Reducing Energy Waste in Network Devices During Idle Periods


Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.
Smart Proxying: Reducing Energy Waste in Network Devices During Idle Periods

Raffaele Bolla *, Maurizio Giribaldi †, Rafiullah Khan * and Matteo Repetto ‡
* DITEN Dept. University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy, Email: {raffaele.bolla, rafiullah.khan}@unige.it
† Infocom s.r.l, P.zza Alessi 2/7, 16128 Genoa, Italy, Email: maurizio.giribaldi@infocomgenova.it
‡ CNIT, Research Unit of University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy, Email: matteo.repetto@cnit.it

Abstract—Although network devices embed low power states as an effective mechanism for reducing energy waste, their use is usually hindered due to the need of maintaining network connectivity for remote access, Instant Messengers (IMs), VoIP clients and other Internet-centric applications. To foster wide usage of low-power states, a low-power entity called Network Connectivity Proxy (NCP) can be designed that maintains the network presence for a set of sleeping devices and wakes them up only when their services/resources are required. This demo paper briefly presents our NCP design and a set of generalized functionalities implemented.

Index Terms—Green networking, energy efficiency, network connectivity proxy, power measurement.

I. INTRODUCTION

Low power silicon technologies are not enough to effectively curb the future network energy requirements. To disruptively boost the network energy efficiency, ad-hoc mechanisms are needed to explicitly manage energy saving by dynamically adapting the usage of the network resources to the actual traffic and service requirements. These objectives have fostered the establishment of the ECONET (low Energy Consumption NETworks) initiative, a collaborative project in the scope of the European Union Funding Program 7 (FP7) aimed at rethinking and re-designing wired network equipment and infrastructures towards more energy-sustainable and eco-friendly technologies. The activities of the ECONET project includes the design and evaluation of the Network Connectivity Proxy (NCP), a low power entity that impersonates sleeping devices by generating/responding to applications/protocols specific messages on their behalf [1]. In fact, the NCP is an effective approach to reduce network energy waste by 60-70% [2].

To maintain presence of sleeping devices, the NCP should preserve their reachability (e.g., support ARP requests and DHCP address renewal), manageability (answer control/management protocols such as ICMP, IGMP etc), application reachability (respond to new connections and packets addressed to transport-layer listening ports) and application state (keep alive transport connections and sending applicationspecific heart-beating messages) [3]. The devices should register desired proxying requests for different applications and protocols with the NCP before entering into low power state. The NCP then sniffs and processes packets intended for sleeping devices and takes appropriate actions [3]. This demo paper briefly presents our NCP architecture and core functionalities of different components.

II. PROTOTYPE DESCRIPTION

The reference architecture of our NCP prototype is depicted in Fig. 1 that is composed of six main components: (i) network traffic filter, (ii) packet processor, (iii) database of rules, (iv) registered hosts, (v) communication protocols and (vi) wake-up mechanisms. One basic requirement for the NCP service is to have up-to-date information about registered hosts, their power states and requested proxying rules. Rules specify the actions that NCP takes for each received packet e.g., respond, discard, buffer, wake-up etc. The registered rules are activated only when the host switches into low-power state and deactivated upon wake-up. The job of the network traffic filter is to sniff and identify packets intended for sleeping hosts based on the header information and pattern matching. The packet processing block determines the appropriate action (wake-up host, send reply packets, send periodic heartbeat packets etc) for each received packet based on the registered rules by the covered hosts. The wake-up mechanisms specify wake-up techniques e.g., Wake On LAN (WOL). The communication protocols provide secure, reliable and flexible communication between the NCP and the hosts. It particularly deals with the rules registration and notification of host’s power state changes [3].
We divided the NCP rules into three sub-categories as depicted in Fig. 1: (i) Application connectivity maintenance which impersonates the running applications state by generating/responding to application specific heart-beat messages, (ii) Network connectivity maintenance which proxy basic network presence and management protocols, (iii) Hosts wake-up policies which specify different conditions for waking-up sleeping hosts. Each rule is composed of four important parts: (i) host identification based on Universally Unique Identifier (UUID) and/or IP/MAC address, (ii) power state that designates the condition to activate a particular rule, (iii) filter specifications based on packet header and/or data fields and (iv) intended action. At present, our NCP design implements following rules: (i) responding ARP request packets, (ii) responding PING request packets, (iii) DHCP address renewal, (iv) Wake-on-Connection (WoC) that wakes-up the host when connection attempt at specified protocol and port is received, (v) Wake-on-Packet (WoP) that wakes-up the host when received packet matches specific pattern, (vi) Send-Reply-on-Packet (SRoP) that sends a given reply packet when received packet matches specific pattern. Our NCP design classifies actions into four types: (i) Wake-up the host, (ii) Send predefined packets, (iii) Build packets and (iv) Build packets by templates (NCP uses templates supplied by hosts). Table I summarizes the basic and generalized implemented NCP rules.

Traffic diversion is also an important prerequisite that enables NCP to get access to packets intended for sleeping hosts. Traffic diversion in LAN environment simply implies binding the NCP’s MAC address to the IP addresses of sleeping hosts. This objective is achieved by sending "Gratuitous ARP", an unsolicited ARP message holding IP address and its associated MAC address. This process replaces host’s MAC with NCP’s MAC in the ARP caches of local network devices during host’s sleep period and vice versa when it wakes-up.

### III. Prototype Implementation

We developed the NCP prototype suitable for home or Small-Office/Home-Office (SO-HO) environment that supports rules specified in Table I. Our software also embeds packet buffering capability to give enough time for sleeping devices to restore their low power states and avoid any packet loss. Our NCP can operate in on-board, switch/router and third party proxying scenarios addressed in [2]. The Universal Plug and Play (UPnP) based communication interface was developed for exchange of messages between the NCP and hosts. The UPnP protocol provides auto-discovery and seamless reliable communication between the NCP and hosts for registration, deregistration, power state notifications etc. The NCP packet filtering block is based on the Pcap libraries, a widely used open source product. The NCP software is developed in C++ programming language with the support of boost libraries. The UPnP functionality is based on the open source portable UPnP (pupnp) libraries.

### IV. Test-bed Setup & Demonstrations

The test bed will be composed by a set of devices as shown in Fig. 2: (i) a PC supporting NCP functionality (NCP Device), (ii) a set of PCs acting as NCP clients (e.g., work stations, video servers, network attached storage devices etc) , (iii) a PC acting as Third-party host trying to access sleeping NCP client, (iv) a network hub/switch, (v) watt-meter to measure the energy consumption of the hosts and evidence of their power state.

A number of experiments will be demonstrated to represent the correct behavior of NCP in two realistic scenarios: (i) Third-party host trying to access sleeping NCP clients from the same LAN, (ii) Third-party host trying to access sleeping NCP clients from outside the LAN. The objective of the demo is to evaluate the effectiveness of UPnP based communication protocol in terms of auto-discovery, latency, network traffic overhead, packet loss etc. The tests will be conducted in both scenarios to exploit all NCP functionalities and rules described in Table I. Furthermore, during each test, the watt-meter will provide the evidence of the actual power state of various network devices.

### ACKNOWLEDGMENTS

This work was partially supported by the European Commission in the framework of the FP7 ECONET (low Energy CONsumption NETworks) project (contract no. INFSO-ICT-258454).

### REFERENCES


---

**Table I**

<table>
<thead>
<tr>
<th>Type</th>
<th>ARP</th>
<th>Ping</th>
<th>DHCP</th>
<th>MAC</th>
<th>IP</th>
<th>SMTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Connectivity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Packet Management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up the host</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send predefined packet</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send packet</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build packet by template</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.** Test-bed & measurement tools.