

Published in:
Proceedings of 2013 24th Tyrrhenian International Workshop on Digital Communications - Green ICT (TIWDC)

Document Version:
Peer reviewed version

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

Publisher rights
© 2013 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.

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, Piazza 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—A network connected host generates/responds to routine applications and protocols heart-beat messages and is considered as disconnected when it fails to do this. Past studies have revealed that about 60-70% people leave their computers powered up 24/7 in the offices and at homes just to maintain the network connectivity. Huge amount of energy savings are possible if these computers can sleep during idle periods but still maintaining their network presence. Thus, a Network Connectivity Proxy (NCP) can be designed which is a software entity running on a low power network hardware such as switch or router and maintains the network connectivity for high power devices during their sleep periods. This paper describes the design, implementation and evaluation of a cooperative NCP that maintains the basic network and application level presence for the sleeping hosts. It also addresses the generic architecture of our NCP and its client softwares.

Index Terms—Green networking, energy efficiency, network connectivity proxy, power measurement, Universal Plug and Play.

I. INTRODUCTION

The energy consumption by computers has become a significant portion of worldwide energy consumption and is continuously growing by 7.5% every year due to their high penetration rate in the world [1], [2]. A recent study by Lawrence Berkeley National Laboratory (LBNL) stated that about 60% of people leave their PCs powered-up 24/7 in their offices only to maintain the network connectivity for Instant Messaging (IM), remote access, Voice-over IP (VOIP) clients, and other administrative management reasons [3]. Thus, a Network Connectivity Proxy (NCP) can be designed that will maintain the presence on behalf of sleeping hosts by generating/responding to their routine application/protocol specific heartbeat messages [4], [5], [6].

This paper presents our extended work on the design and implementation of NCP described in [7]. Our approach is based on cooperative NCP, a more flexible design that allows network hosts to register different rules and actions at the NCP before sleeping [8]. The network hosts can specify how NCP should respond to specific packets or periodically generate/respond to applications/protocols specific heartbeat messages. The NCP also allows the network hosts to specify different conditions upon which it has to wake them up. The NCP uses traffic diversion mechanism to access the packets intending for sleeping network hosts and processes them based on their registered rules. This paper presents the basic NCP requirements and describes its internal architecture in details. It addresses how TCP keep-alive feature can be used to preserve open TCP connections on behalf of sleeping hosts. Further, it also describes the design of NCP client’s application and communication module used for exchange of information between the NCP and its client devices.

The rest of the paper is organized as follows. Section II presents a brief overview of the NCP. Section III describes in details different architectural components of our NCP design. Section IV presents the basic structure of NCP client’s application. Section V evaluates the NCP performance. Finally, Section VI concludes the paper.

II. OVERVIEW OF NETWORK CONNECTIVITY PROXY

The NCP maintains network presence of hosts by acting on their behalf [4]. Its basic requirements at different layers of TCP/IP protocol suite are presented in [8]. Maintaining network presence mainly implies three issues: (i) knowing the power state of devices the NCP is covering, (ii) being aware of traffic addressed to suspended hosts, (iii) representing, maintaining and exchanging service state for various protocols and applications. To cope with above issues, the NCP must process packets addressed to sleeping/suspended hosts and carry out specific tasks according to hosts’ requirements. Simply, NCP preserves host reachability (e.g., ARP, DHCP lease, NetBIOS etc), host manageability (responding control/management protocols as ICMP and SNMP), application reachability (allowing new connections and responding packets addressed to transport-layer listening ports) and application state (keeping alive transport connections and sending application-specific heart-beating messages) [8].

Different system architectures can be envisioned to implement the NCP service. The first architectural choice is between invisible or cooperative NCP [9]. The invisible NCP does not advertise itself; hosts are unaware of its presence and thus no interaction takes place. The NCP must inspect all traffic to/from the covered hosts and thus should be placed according to this requirement (e.g., switch or a home gateway). Further, it cannot detect if the host permanently leaves the network. Thus, our work focused on the design and implementation of cooperative NCP which requires interaction with covered...
devices. Cooperative NCP offers large flexibility in terms of desired behavior for sleeping hosts and in controlling when operations should start and stop. It requires an application running on the hosts that is used for communication with the NCP. The second architectural choice is the location of the NCP service [8]. Our NCP software is capable to work in the following scenarios.

1) On-board scenario: The NCP software can be embedded on host’s NIC. This approach brings some benefits: e.g., the NCP shares same IP and MAC addresses as the host itself, and mobility is easy to manage as NCP moves together with the host. The main limitation here is the low NIC’s memory and processing capabilities. However, future NIC designs might have enough storage and processing capabilities, thus allowing implementation of a complete NCP service covering many applications and protocols.

2) Switch/Router scenario: The NCP service can also be placed on an always powered up network entity such as hubs, switches and routers. These devices usually have internal CPU and run an embedded OS such as Linux or FreeBSD which eases the deployment of NCP service. The NCP can have enough processing capability and allow development of sophisticated routines maintaining presence for large number of applications and protocols. Further, routers/switches provide hardware accelerators for packet filtering and classification which can greatly simplify the most resource-starving tasks of the NCP. Another consideration is the ability to see the traffic addressed to the sleeping hosts: that is likely in Small Office/Home Office (SOHO) environments where usually one device provides connectivity to the whole site but cannot be taken as granted in multi-homed scenarios. This scenario also requires a communication protocol for information exchange between NCP and its clients.

3) Standalone scenario: The NCP service can be provided by some host other than network equipment. From the perspective of energy balance, this is the most critical solution, as the standalone device must consume far less power than the saving from average number of sleeping devices. However, high memory and high computational power would be available, and that would allow implementing complex routines for a large number of applications and hosts. This scenario requires network traffic diversion to access the packets addressed to sleeping hosts. Further, a communication protocol is also required for information exchange between the NCP and its clients.

III. NCP Reference Architecture

NCP requires four main architectural components as shown in Fig. 1: (i) database of rules, (ii) Network traffic filter, (iii) Wake-up mechanisms, (iv) Communication protocol. NCP rules bind the network traffic addressed towards sleeping client devices to specific actions. Network traffic consists of single packets or connections; their identification is mainly made by a set of conditions that take into account header information (source and destination addresses, source and destination ports, protocol, protocol-specific flags and options) or packet content (pattern matching, application-specific headers and information). Actions specify the operation the NCP is expected to undertake. The NCP rules can be divided into three sub-categories: (i) Application connectivity maintenance which preserves the application state on behalf of sleeping hosts by generating/responding to routine application specific heartbeat messages, (ii) Network connectivity maintenance which includes basic network presence and management protocols (e.g., Address assignment and resolution, host location, presence verification, etc.), (iii) Wake-up polities which includes different conditions for host wake-up. Rules are registered by NCP clients prior to their application and are stored internally as long as they are explicitly withdrawn, in order to follow the dynamic needs of client devices. The NCP distinguishes among active and inactive rules: a rule is active when the corresponding device has delegated the NCP to act on its behalf (i.e., when the device is in low power, standby or power off modes), inactive when no delegation has been made (i.e., when the device is in low power or full power modes).

When a device is going to switch to a low power mode, it notifies the NCP, so that its set of rules are activated. Rule activation implies two tasks. First, traffic addressed to the client is diverted towards the NCP. Second, the packet filtering engine is set to pick up packets matching the traffic patterns contained in that rule (e.g., source/destination address, source/destination port, protocol, etc.). The first task is achieved if the NCP lays on the path between each of its client device and all of its peers; although this hypothesis can only be suitable for switch/router scenario. Thus, network traffic diversion should be considered an integrating issue in packet filtering. The NCP packet processor carries out the operation specified in each NCP rule for each packet matching that rule’s filtering specifications. Different kind of operations should be considered, depending on NCP’s processing power and on the complexity of transferring application/protocol status between the client devices and the NCP. The NCP should account for at least three kinds of operations: waking up devices, replying to incoming packets and sending unsolicited packets. Starting from these basic types, a wide variety of more complex and specific operations may be derived.

Depending on the communication medium, NCP uses different type of host wake-up mechanisms e.g., Wake On LAN (WOL), Wake on Wireless LAN (WoWLAN) etc. Suitable
communication protocols are needed for managing communication between the NCP and its clients. They should deal with registration of rules and notification of changes in the power state of devices. Useful requirements for such protocols include automatic discovery of NCP and its clients, fault tolerant and resilient communication, minimum overhead, secure and trustworthy operation.

A. NCP rules layout

Rules are the description of the behavior clients ask the NCP to undertake on their behalf. Each rule is made of four parts: the host identification, the power state, a filter specification and an action. A better option for host identification is using a Universally Unique IDentifier (UUID) derived from some invariant host characteristic. The power state indicates when the rule should be applied. The NCP activates a rule when the device switches to a power state equal or lower than that held by that rule. The filter identifies packets for which the rule applies. It is made of a set of conditions, which are logically AND-ed to give the pattern matching criteria.

1) ARP rule: The ARP rule is unavoidable for NCP operation; thus it is automatically activated when clients go to sleep. It instructs the NCP to answer ARP requests on behalf of the sleeping host. In theory, the host’s IP address should be resolved to the host’s MAC address. However, packets sent upon the resolution will be delivered to the sleeping host and this could be an issue if the NCP does not lie on the path between the two hosts. Thus, the most general and flexible solution is to provide the NCP’s MAC address in ARP responses. This approach brings two issues. First, some hosts on the LAN may already have gotten the MAC address by a previous resolution when the host was awake; in this case, their traffic would be sent directly to the sleeping host and could not be seen by the NCP. Second, when the client wakes up, it must take back the management of its network presence, and that implies getting the correct traffic delivery again. Gratuitous ARP solves both issues, and is therefore used together with ARP to implement the ‘traffic diversion’ (see Section III-C).

2) Ping rule: The Ping rule instructs the NCP to answer ICMP echo request messages on behalf of the sleeping host. ICMP echo requests can carry additional data in their payload. The NCP should copy this data without any modification in the echo reply packet it generates.

3) DHCP rule: The DHCP rule instructs the NCP to preserve the sleeping hosts IP by periodically sending DHCP lease renewal request to a DHCP server.

4) Wake on Connection rule: The Wake-on-Connection (WoC) rule instructs the NCP to wake up the sleeping host when packets addressed to a given transport port are seen. Here the term ‘connection’ is interpreted in a broader sense which means an attempt to establish a dialogue with the sleeping host. Thus, the WoC rule is used both for connection-oriented and connectionless protocols (e.g. TCP and UDP). The packets that trigger the WoC rule should be delivered to their actual destination host. The NCP solves this issue by buffering such packets and sent them out once their target device has completely woken up.

5) Wake on Packet rule: The Wake-on-Packet (WoP) rule instructs the NCP to wake up the sleeping host when packets matching a given pattern are seen. That could be used for every type of packet, but in practice it would be useful especially for connectionless datagrams (i.e., UDP packets). The logic behind this rule is to find some application-specific bulk (pattern) of data within the packet which are known a priori. The remote peer may send several packets to improve reliability; however, these packets are often sent sequentially and likely will all find the destination host still sleeping. In this case, the NCP buffering is even more useful than with the WoC rule.

6) Send Reply on Packet rule: The Send-Reply-on-Packet (SRoP) rule instructs the NCP to send a given reply when packets matching a given pattern are seen. Unlike this rule can apply to connection-oriented protocols, as of the need to keep and update a connection status both internally and in packet headers. This rule uses the same filtering logic as the WoP rule. In this case, when a packet matching the pattern is found, a reply packet is build with the data contained in the SRoP rule, and this packet is sent back to the source of the incoming one.

B. Basic NCP actions

The second part of a rule is the action the NCP is requested to carry out. Differently from the filter, the action is often implicit in the rule type, thus there is no data to be included during the rule registration with the NCP. There are five kinds of actions our NCP implements:

1) Wake up devices: The NCP is expected to do nothing but waking up devices when any packet matches the registered rules.

2) Sending predefined packets: This is possible if the reply to an incoming packet or an hear-beating message is foreseeable and does not depend upon any parameter. The client devices can create the body of these packets in advance and load them at the time of rule registration while the packet headers should be added by the NCP.

3) Building packets: The NCP understands the protocol/application specified by the rule and creates packets on its own to reply incoming requests or for periodic heart-beating.

4) Building packets by templates: The NCP builds packets starting from templates supplied by client devices; templates are filled in and modified according to a set of patterns and instructions contained in the NCP rule.

5) TCP keep-alive: The NCP answers the TCP connection keep-alive messages while the host is sleeping. It is mainly conceived to maintain TCP connectivity while no data have to be exchanged between the peers. Three behaviors are possible when data is sent on the connection: (i) wake up the host, (ii) advertise a zero-window condition to delay data reception from the remote side, (iii) buffer packets and acknowledging them
C. Network Traffic Diversion

One of the prerequisite for NCP operation is the capability to access packets intended for sleeping devices. If the network topology forces all traffic to cross a common point, placing the NCP there would automatically meet the above requirement. In realistic scenarios, that would require single-homed sites; traffic among internal hosts should traverse that device as well. Traditionally, the home environment reflects this scenario: one or more devices are connected through a bridge integrated in the access router. However, this approach has few limitations: (i) The number of bridged ports on low-end access routers is usually very small (typically 1 to 4), thus many users install additional bridges and access points to connect more devices in their home network. (ii) Many hosts have multiple NICs, that could be active concurrently. (iii) If the traffic volume is high, filtering could slow down packet forwarding operations, unless costly specialized hardware is used. Thus, traffic diversion can be the optimal solution which forces network devices in the local network to forward packets intended for sleeping hosts towards the NCP.

In LAN environment, the Address Resolution Protocol (ARP) broadcasts request messages including the IP address of the destination and asking for its associated MAC address; the destination then answers providing its MAC address. ARP resolutions are stored in caches by each host and removed after a time-out occurs. The network traffic diversion is quite simple if using ARP protocol for traffic diversion. It is just a matter of binding the NCP’s MAC address to the IP addresses of sleeping hosts. When a device falls into a sleep state, the NCP’s MAC replaces the host’s MAC, and when the device comes out of the sleep state, the host’s MAC is brought back. Two actions must be implemented for this purpose: (i) The NCP must answer ARP requests on behalf of sleeping hosts providing its own MAC address, (ii) The NCP must update ARP caches of local hosts when its clients enter or exit the sleep state. This action is necessary to avoid sending packets to obsolete MAC addresses in caches. This kind of traffic diversion requires sending a ‘Gratuitous ARP’ packet which is an unsolicited ARP request message holding the IP address and the associated MAC address.

D. Communication Protocol

The communication protocol is responsible for providing flexible, efficient and reliable communication between the NCP and its client devices. Our design is based on the Universal Plug & Play (UPnP) protocol that enables zero-configuration, transparent networking and automatic discovery for networked devices. UPnP allows devices to announce their presence in the network, to discover other devices, to get descriptions of available devices and services, to control remote services, to subscribe for event notification, to get high-level human-readable presentations of devices and services [10]. There are two roles in the UPnP architecture: Controlled Device (CD) and Controlled Point (CP). CDs are like servers that implement one or more ‘services’ and periodically advertise their presence over the network. The service is a logical functional unit that receives instructions from the CP, performs some actions and notifies the CP about the changes in state variables. The CP retrieves the controlled device and its services description, invokes actions implemented by the services, calls service state variables and receives event notifications from the services [7].

The UPnP architecture allows the NCP clients to automatically discover the NCP location in the network, to register to the ‘NCP service’, register required actions, and to smoothly leave the network [7]. Fig. 2 shows the basic UPnP model for the NCP service. Besides managing basic network presence and management protocols (e.g., ARP, PING, DHCP etc), the NCP service also offers its client devices to register WoC, WoP and SRoP actions. However, this simple UPnP model for building the NCP service can be extended to account for more functionalities. Apart maintaining network presence, the NCP could also manage the power state of devices, so to put them to sleep when no traffic is present and to wake them up when their processing is required. In this case, the NCP should be aware of devices present in the network and of their Low-Power (LP) capability and state. The UPnP infrastructure can be used for this purpose as well, but now the UPnP CDs are the hosts that offer ‘LP service’ inside the LAN and CP is the NCP [7]. With this approach, both the NCP and its client devices play the role of UPnP CD as well as CP, but for different UPnP services, as shown in Fig. 3.

IV. NCP Client’s Application

The NCP client’s application is responsible for providing effective interface for the hosts to register their requests at the NCP. The client’s application also enables the NCP to get access to the LP service of its client’s UPnP device. This will
make possible for the NCP to know about the power state or schedule sleeping of its client device during the specified periods. The generic architecture of the application for the NCP clients is shown in Fig. 4. Its main components are briefly addressed below.

1) User Interface: It provides basic application control options for the user. It also provides interface for the user to register required actions at the NCP service.

2) HostManager: It contains the information about the host’s current status and allows the host to send/receive the actions requests from the NCP. It is responsible for managing both, the host CP and CD.

3) Control Point: The CP enables the host to register actions at the NCP service. These actions register basic network/application level presence requests for the host during sleeping period and also define the conditions for wake up.

4) Controlled Device: It implements the LP service and contains information about the host power status. The service contains actions to manage the host power state. The control server handles basic service management and service/actions registration requests. The CD through eventing keep the NCP updated about the host state table.

5) PowerSavingModule: It is responsible to keep the host CD’s state table updated about the current power state. It continuously checks if the OS kernel receives any kind of power state change request. It modifies the corresponding state variables value in the CD which immediately informs the NCP’s CP about the change.

V. EXPERIMENTAL EVALUATION

This section checks if the NCP works in various realistic scenarios and measures its impact on network operation due to host power state transition delay, packet loss, NCP latency, traffic overhead etc.

A. Measurements from Test-Bed

The tests were performed considering two realistic scenarios based on the location of third-party host: (i) Scenario 1: Both, the NCP client and third party host lie in the same local network. (ii) Scenario 2: The third party host tries to access sleeping client from outside the local network. The main objective of scenario 1 is to check if the NCP can maintain the sleeping hosts presence inside the local network while scenario 2 checks if sleeping hosts can be accessible from outside the local network. Activating/De-activating traffic diversion through gratuitous ARP takes a bit of time during which some packets addressed to sleeping hosts may be lost. The effect of this delay is roughly evaluated by sending PING messages and counting lost PING responses during the host’s power state transition. The average values are shown in Table I during wake-to-sleep transition (WTS) and sleep-to-wake transition (STW) in both scenarios. Few PING responses are lost in scenario 1; while no packets are lost in scenario 2. The main reason behind this is the delay produced by traffic diversion in updating the ARP caches of network equipments in scenario 1. While scenario 2 don’t require traffic diversion as the traffic already passes through the NCP. Similar considerations also hold for PING duplicate responses. No duplicates are received in scenario 1 as the traffic diversion requires some time after the host wakes up. While in scenario 2, the host needs to inform NCP about its wake-up to stop proxying. This takes some time and few PING packets are responded by both, the NCP and the host.

Many applications usually wait for few seconds to receive response before the connection is aborted. The NCP clients should wake up and answer the incoming packets before such timeout occurs. To avoid packets loss, the NCP buffers the incoming packets and transfers it to the host after wake-up. The latency for waking up sleeping host includes: (i) NCP latency to send Wake-On-Lan (WOL) packet, (ii) Time taken by host to resume from sleeping state and notify NCP about it. This was evaluated for WoC rule using the Secure Shell (SSH) remote connection protocol. The average values are shown in Table I for both scenarios. NCP latency is negligible and is measured as the time elapsed from receiving SSH request to the time when the WOL packet is sent. SSH response time is measured as the time when third party host sends SSH request to the time when it receives the SSH response. While the host wake-up time is measured as time elapsed from sending WOL packet and first update packet sent by the host to the NCP.

B. Traffic Overhead

The test was performed for a duration of 712 seconds during which the NCP’s client device stays in steady state (periodic presence advertisements) for a while, gets registered with NCP, registers a WoC action, transitions into sleep and wake-up mode and finally de-registers from NCP. The test results show that 1.33 packets per second were exchanged having average size of 145 bytes. The network overhead during each individual phase is shown in Table. II. It can be observed that about 52% of the total packets were exchanged during steady state condition while the average packet size is the smallest. Steady state condition contains only periodic presence advertisements which are quite small in size. About 26% of packets are exchanged during discovery and client registration but the average packet size is quite large as
both, the NCP and its client download the XML based UPnP description file of each other. While few small size packets were exchanged during action registration and power state notification.

C. Expected Energy Savings

The real energy savings that can be achieved by the adaptation of NCP depends on the number of NCP clients in the world, their active and sleep power requirements and their average use per day [8]. Fig. 5 shows the global expected energy savings that are calculated using Energy Star office equipment calculator by considering 815 million Internet connected desktop PCs and 164 million Internet connected laptops. It considers 6.5 hours as average PCs usage per day. The world-wide energy savings will account to approximately 378 TWh/year which is equivalent to 83 Billion Euro savings/year considering 22 cent/kWh as the average cost of electricity in Europe.

VI. CONCLUSIONS

The NCP is an optimal strategy for reducing the network energy waste. It allows network hosts to sleep as long as their resources are not required and maintains presence on their behalf. This paper has addressed the basic NCP requirements and its possible architectural designs. It mainly focused on the NCP and its client devices software architectures. This paper also described the design of a flexible communication module based on UPnP protocol that provides auto-discovery and seamless communication ease. Some preliminary tests were performed to check the correctness of basic NCP functionalities and to evaluate its performance in terms of packet loss and network overhead. The UPnP based NCP design is also quite useful to easily extend proxying capabilities for other network devices such as printers, scanners, copiers etc. In short, the NCP is quite useful approach that has potential of providing billions of Euro savings every year.

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