4

Energy Efficiency of Connected Mobile Platforms in Presence of Background Traffic

Sameh Gobriel, Christian Maciocco and Tsung-Yuan Charlie Tai
Circuits and Systems Research Lab, Intel Labs, Intel Corporation
USA

1. Introduction

In the last decade there has been an explosive growth in popularity of mobile computing platforms which include laptops, notebooks, tablets, cell phones, etc. However, the usability of these devices from end users point of view is directly associated with their battery life and the fact that they are powered by non-continuous energy sources imposes a serious limitations to these devices. As a result, a typical architecture design of such mobile computing platforms will include low power platform states for individual components (e.g. CPU, memory controller, hard-disk, etc.) or for the whole platform. These low power states can be *sleep* states where the components or the whole platform is in a non-operational mode (e.g. operating systems defined sleep states: standby, hibernate, etc.) or *scaled* states where they are operating at lower than peak performance (e.g. CPU dynamic voltage and frequency scaling (DVFS)). For example the Advanced Configuration and Power Interface (ACPI) specifies power management concepts and interfaces. ACPI integrates the operating system, device drivers, system hardware components, and applications for power management and defines several power states for each component ranging from fully powered on to fully powered-off with each successive state consuming the same or less amount of power.

Individual power management techniques differ in (1) how deep the low power state is, where usually a deeper sleep state has a higher exit latency and hence a negative performance impact, (2) the algorithms used to manage entering and exiting these low power states and (3) optimizations to extend these sleep states as long as possible so additional system components having longer exit latency could also be put into lower power state. Standard policies are usually based on *timeout* approaches, where the platform or individual components are transitioned in low-power modes if they have been observed "idle" for the duration of a timeout. Usually this timeout value is dynamic and depends on the current operating state and the previous history of idle and active platform states.

Recent measurements and field assessments have shown that: in offices, desktop systems at a minimum remain powered up all day long whether being actively used or not, and further, two-thirds of such systems are fully on after work hours, with only 4% operating in sleep modes. It appears plausible that network connectivity drives much of this and these systems remain available to facilitate sporadic, occasional activity, such as user remote access, administrator access for maintenance, etc. (backups, patch management). On the other hand, in the residential sector, the average PC is on 34% of the time and spends only 4% of the time

in an some form of sleep state and more than half the time a PC is on no one is actively using the machine.

In this chapter we will argue that a major hurdle for end systems desiring to enter a sleep state arises from the network and the desire of end users to stay *"always on and always connected"* to the network. We first focus on powered-on platforms but idle (i.e. not running an active heavy workload) and quantify the effect of background network traffic on the platform energy-efficiency. then, we propose a low-overhead mitigation techniques that can detect and classify network traffic with no dependencies on network protocols nor the platform and present simulation analysis of these techniques and algorithms based on live network traces. Finally we describe our firmware based implementation in our prototype Wireless Network Interface Cards (WNIC) and show our practical results when applied in live networks.

2. Related work

Several mechanisms have been proposed to reduce the energy consumption of networked platforms. Prior work can largely be grouped in three categories: reducing the active power consumption of systems when they are awake Agarwal et al. (2008); Flinn & Satyanarayanan (2004); Li et al. (2007), reducing the power consumption of the network infrastructure, routers and switches Gunaratne et al. (2005); Gupta & Singh (2003); Nedevschi et al. (2008) and opportunistically putting the devices to sleep Agarawal et al. (2009); Shih et al. (2002); Sorber et al. (2005). *Long Idle* falls in the third category, where it advocates for localized energy-efficient optimization within the platform to extend the sleeping state of the platform rather than using a network-wide implementation of a proxy (wakeup) service. However, in contrast with previous work *Long idle* did not assume a special magic packet for wake-on WLan (WoWLAN) Shih et al. (2002) nor it uses multiple network interface cards Sorber et al. (2005), and it did not assume any application level optimization to offload background traffic processing Agarawal et al. (2009). *Long idle* requires only a single network interface card and detects background traffic without the need to decrypt the data payload by gathering traffic statistics (size, direction, interarrival time, etc.) on the ongoing communication. In this chapter, we show that such non intrusive technique achieves a good detection accuracy and extends the platform sleeping state with no need for a more sophisticated technique or a special hardware.

3. Problem formulation

In this section, we analyze and quantify the negative impact of continuously interrupting the platform to process the background and management network traffic and highlight the effect on the platform energy-efficiency.

3.1 Platform PM energy gain

Current platform power management guides the platform to enter a lower power state (sleeping state Sx) if it detects a period of idleness; τ_{idle}. The more the system stays in sleep state un-interrupted the better the energy gain is, because more and more peripherals can enter a low power state and/or because deeper sleep states with longer exit latency can be used.

Recent work has shown that Smart timing approaches for Operating systems (OS) can be used to increase the quiet times for the OS by skipping timer tick interrupts when the platform is idle or by adaptively changing the rate of timer interrupts (e.g., Olsen & Narayanaswami (2006)) which form the basis for *"tickless OS"* where the OS is moving away from scheduling periodic clock interrupts every few milliseconds to a more event-driven approach Gleixner & Molnar (2006); Yodaiken & Barabanov (1997) where the OS is waken to process an event being posted by the applications or by the hardware on demand.

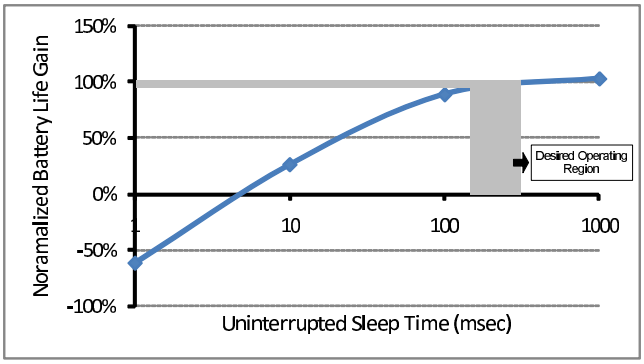


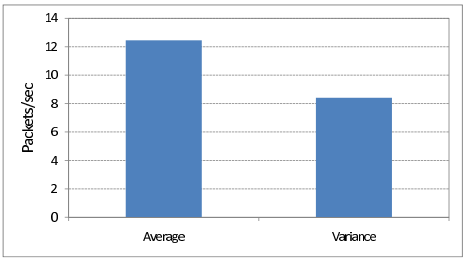
Fig. 1. Typical Platform PM Energy Gain

Figure 1 represents the energy gain of platform power management and typically the achieved gain follows a non-decreasing concave reward function Olsen & Narayanaswami (2006); Siddha et al. (2007). The x-axis represents the break event time, i.e., guaranteed platform quietness period, and the y-axis represents the normalized increase in the total battery life when the system enters a low power state normalized to the case when it remains in the same fully active state (S0). The desired operating range is highlighted and extending the sleeping time beyond the highlighted range has a diminishing return in the energy gain as it is not possible to extend the tick rate for the OS beyond the highlighted range Olsen & Narayanaswami (2006) and the system will be wakened for some scheduled event at about this range.

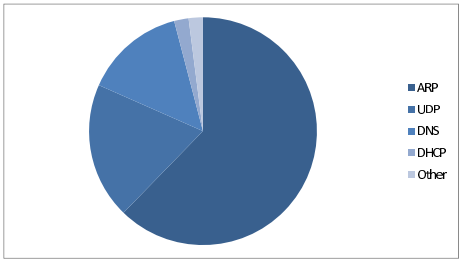
3.2 Background traffic effect on platform PM

When a system is connected to a network, even when the user is not engaged in any active communication the system engages in what we term "background/noise" traffic. These packets can be network maintenance or management packets (e.g., ARP, DHCP, etc.), network services handshake (e.g., directory services, NetBIOS) or application heartbeats (e.g., Instant Messaging, Windows Widgets, etc.). Figures 2 and 3 show the average and variance in the number of network packets processed per second and the protocol mix of two examples of this background traffic. Figure 2 is a wired network trace for a home environment and Figure 3 is a wireless trace for a platform connected to an enterprise network.

It should be clear that these network traces are shown here as examples of real-life networks and cannot be generalized as the default background traffic in these networks. As previously mentioned the traffic characteristics will be remarkably different from location to location based on the network infrastructure architecture, platforms and applications used, number of network users, time of data collection etc. In fact chattier and less chatty background



(a) Traffic Rate



(b) Protocol Mix

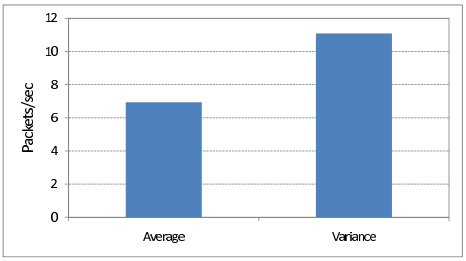
Fig. 2. Home Wired Network Background Traffic

and management network traffic has been reported in different literatures Gunaratne et al. (2005). One of the significance of our work is that it is not designed to target specific network conditions and no specific background/management traffic network optimization (e.g., ARP-Proxy Nedevschi et al. (2009), Network Proxy Offload *TC38-TG4 Proxying support for sleep modes Specifications* (2009), etc.) is assumed.

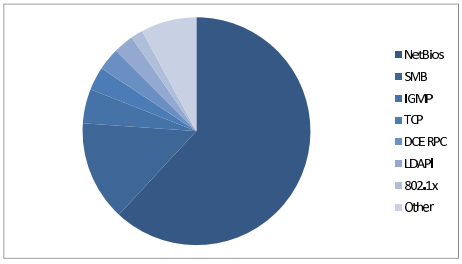
For a networked platform keeping the sleep state of the system uninterrupted is a challenge. Network activity will interrupt the sleeping state because the host is waken-up to process these packets. Moreover, each time the sleeping state is interrupted the platform will have to wait for the timeout (i.e., τ_{idle}) before re-entering the low power state.

It should be noted however, that τ_{idle} can't be a small value because this will have a negative impact on active workload due to the overhead (entry and exit latency and power overhead) associated with transitioning to and from the low power managed state. For example if a Wireless NIC is used and according to the 802.11 Power Saving Mode (PSM) *IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications* (1999) each time the NIC enters the sleep mode, it has to wait for the next beacon advertised by the base station before it can receive any of its buffered packets in the base station. Typically the beacon interval is set at 100 msec which means that the exit latency from the sleep state of the wireless NIC in PSM is 100 msec. Hence, if the sleep state is entered blindly each time there is packet-free time the active communication performance (e.g., throughput, latency, etc.) will significantly degrade.

The net effect of background traffic can be viewed as depicted in Figure 4 (dotted red line). If the interpacket arrival time is X msec then longer platform sleep times are not available and thus the energy gain of platform power management beyond some value is not achievable.



(a) Traffic Rate



(b) Protocol Mix

Fig. 3. Enterprise Wireless Network Background Traffic

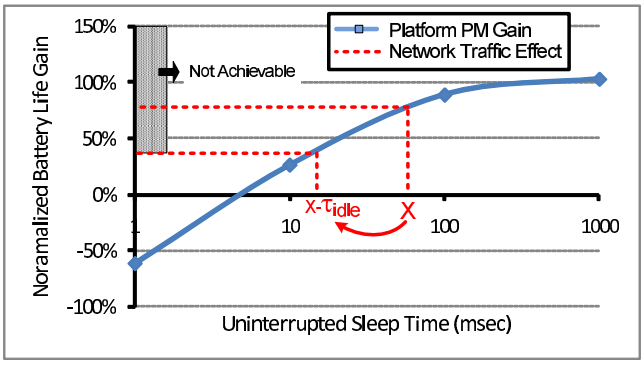
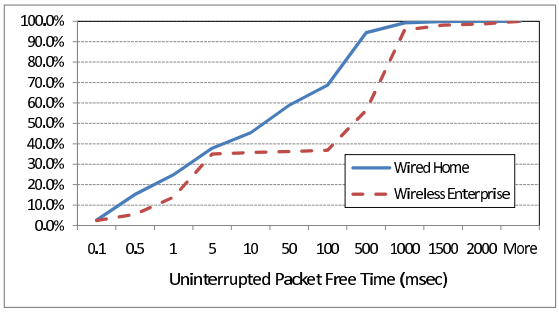


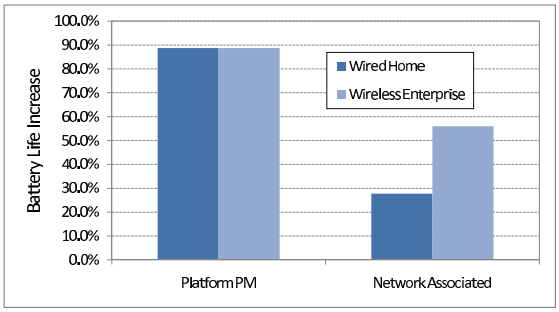
Fig. 4. Traffic Effect on PM Energy Gain

Moreover, encountering an additional performance guard timeout before re-entering the low power state will degrade the net energy gain of platform power management even further (for illustration $\tau_{idle} = 50$ msec in figure).

Figure 5(a) plots the Cumulative distribution function of uninterrupted packet free times in different environments. As shown in figure in an enterprise environment (traffic trace example shown in Figure 3) almost 40% of the time the platform will be hit by a packet within 100 msec and in a wired home environment (traffic trace example shown in Figure 2) the probability is about 60%. The platform power management energy gain degrades accordingly with background network traffic. If we assume that the platform activities are synchronized and an uninterrupted sleep time of 100 msec is guaranteed each time the platform enters



(a) CDF of Packet Free Time



(b) Energy Gain @ 100 msec

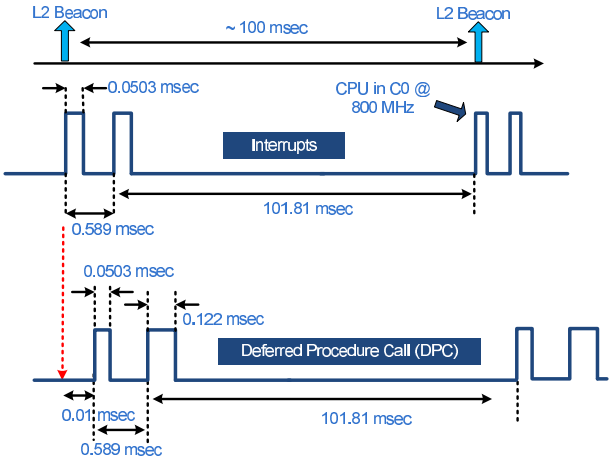
Fig. 5. Platform PM Gain With Network Traffic

the sleep state then battery life is extended by almost 88%. On the other hand when the platform is associated with the network and is processing the background network traffic the gain diminishes (even with $\tau_{idle} = 0$ msec) to only 56% for the example wireless enterprise network and to about 27% for the wired home environment as shown in Figure 5(b)

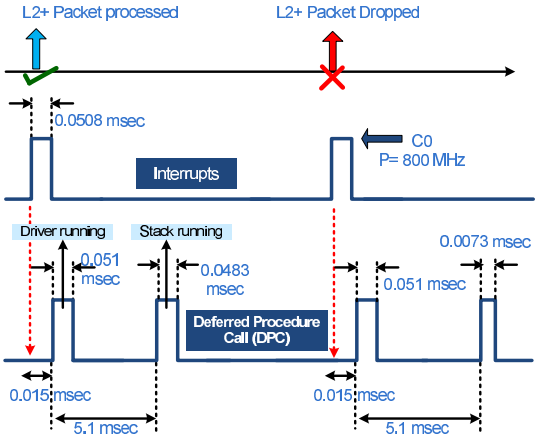
3.3 Background traffic processing cost

We use Windows Performance Tools (WPT) Kit *Windows Performance Analysis Tools* (2008) designed for measuring and analyzing system and application performance on Windows to quantify the processing cost of each packet of background traffic at the different layers of the networking stack and to highlight that each packet will generate at least one interrupt which will wake the platform up to exit the low power platform state.

Figure 6(a) plots the processing cost of Layer 2 (L2) packets. It shows the interrupts and the Deferred Procedure Calls (DPC) generated by the NIC when L2 packets are processed. As shown in the figure when beacons are received the NIC generates two interrupts reflecting information (TIM info, SNR data, etc.) being DMA-ed into the host memory these two interrupts are spaced within approximately 0.5 msec. Each interrupt is followed by a DPC reflecting that the NIC driver is running. Investigating the shape and reasons of these interrupts is beyond the scope of this chapter but we highlight that for each of these interrupts or the DPC the CPU is running (C0 at 800 MHz) indicating that host is wakened up to process each of them.



(a) Cost for L2 Packets



(b) Cost for L2+ Packets

Fig. 6. Processing Cost for Background Packets

Figure 6(b) plots the processing cost of L2+ packets (all the packets that are traced in Figures 2 and 3). As shown in figure each of the L2+ packets cause an interrupt reflecting a memory DMA for the packet data followed by two DPCs one is caused by the NIC driver running and the other by the TCP/IP OS stack running. The running time of each depends on whether the packet is dropped or is processed but similarly what we want to highlight is that the host is interrupted at least once to process each of these packets.

4. Long Idle technology

We define an architecture and a low-overhead technique, which we call *Long Idle* to enable the NIC to classify incoming packets into active versus background traffic and intelligently hold the background traffic on the NIC device to guarantee idleness so that the system can

enter low power states with no impact on performance. It should be noted, however, that although we present *Long Idle* technology as something that we implemented in the NIC there is nothing that prevents the network designer from applying that same technique into for example, network switches and/or network wireless access points.

4.1 Long Idle overview

By definition background packets are not for active communication, hence, typically these packets are not time critical and can be delayed for extended period of time without any impact on the user experience. In fact most of these background packets are discarded by the host after processing them Nedevschi et al. (2009; 2008).

Unfortunately, at the NIC level it is not possible (without performing deep inspection of the packet) to know whether the packet will be of any use to the host or not. Since for all practical reasons the time and processing overhead and energy of deep packet inspection is intolerable at that NIC level if at all possible due security and encryption (e.g., VPN, etc.), the hypothesis of *Long Idle* is: if the NIC can classify the incoming packets into active versus background traffic without having to look inside the packet then the NIC can intentionally buffer the background packets and guarantee uninterrupted quietness to the platform for extended amount of time, e.g., few hundreds msec, and then send these packets as one burst to the host for processing. Figure 7 depicts the *Long Idle* concept. Moreover, since the created quietness period is guaranteed not to be interrupted except for active traffic then the platform can *instantly* enter a sleep state for the whole duration of the buffering period (without waiting for τ_{idle}) to save its energy even further.

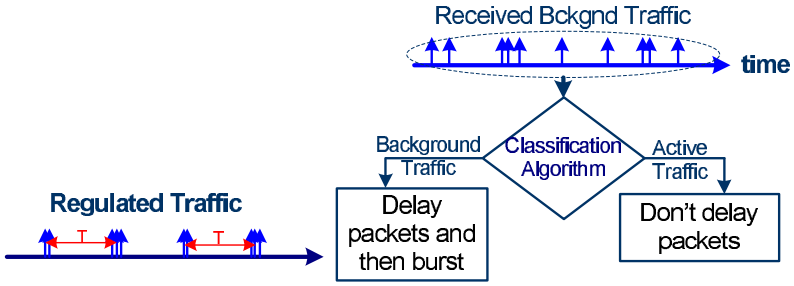


Fig. 7. Long Idle Overview

4.2 Long Idle algorithm

Once a packet is received at the NIC and is classified as belonging to background traffic the packet is not instantly pushed to the host, but instead buffered inside the NIC's internal memory until one of few events occurs: Either, a *Long Idle* timeout event has occurred (maximum time watermark (e.g., 300 msec) for holding a background packet is reached) or a packet belonging to an active communication is received in the Rx FIFO or the Rx FIFO has reached some occupancy threshold (maximum bytes to hold for background traffic).

Long Idle relies on the NIC's ability to differentiate the active traffic from the idle background one without a need for packet deep inspection. Based on our analysis the main differences in the traffic patterns can be summarized as follows:

1. Idle traffic is mostly small-sized packets (typically a heartbeat or a management packet is only few tens of bytes) and MTU size packets are rarely seen in idle traffic.
2. Because most of the management and background traffic is transmitted by the network, idle traffic is mostly on the receive path.
3. Active small-sized packets (e.g., those belong to a VoIP or a gaming session) are mostly two-way communication.
4. A large percentage of the Idle traffic is broadcast packets, while active communication is mostly sending and receiving unicast packets.

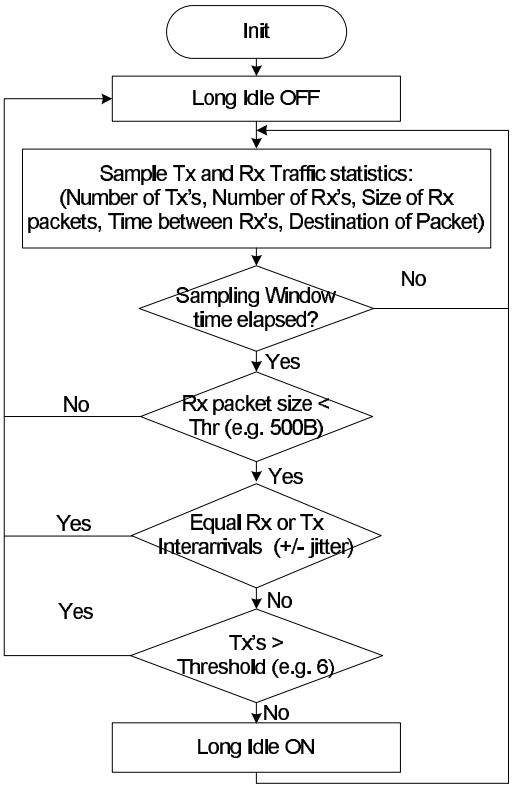


Fig. 8. Long Idle Sampling State Machine

Our algorithm for enabling *Long Idle* is composed of two state machines. One is continuously sampling the ongoing traffic and classifies whether it is active versus background at the end of the sampling window and is independent from the NIC used (wired or wireless), the other is defining how to handle the packet based on the classification result and clearly this depends on the type of the NIC used. In this chapter, we use the Wireless NIC as an example and show how the packet handling can be coupled with 802.11 PSM u-apsd *IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications* (1999) mode.

As shown in Figure 8, the algorithm for traffic sampling and *Long Idle* divides the time into slices of sampling windows. During this sampling window statistics on the size,

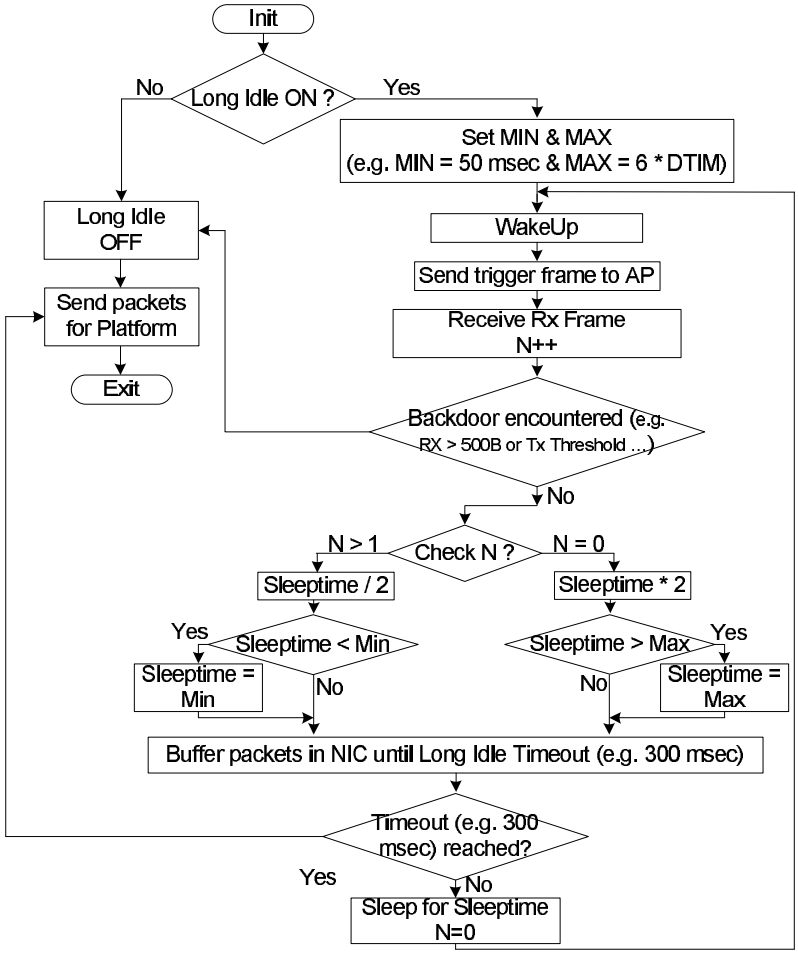


Fig. 9. Long Idle WNIC Implementation

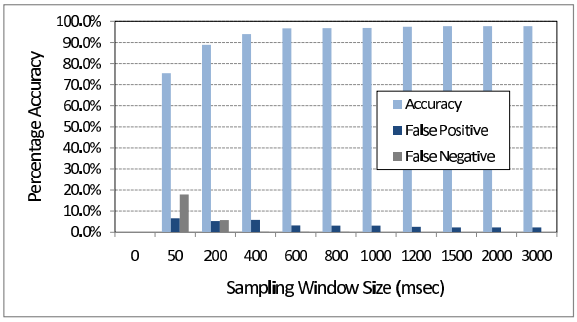
destination, interarrival times, and number of transmitted and received packets are collected and based on the conditions outlined above the ongoing communication is classified as active versus background one. Figure 9 depicts the wireless NIC sleep time update based on the classification of the ongoing traffic. If *Long Idle* is detected the Wireless NIC sleeps longer and the received packet is buffered for extended amount of time. On the other hand, as soon as the NIC receives a packet that it classifies as belonging to an active communication the *Long Idle* mode is set to off and these packets are pushed to the platform instantly.

5. Evaluation and analysis

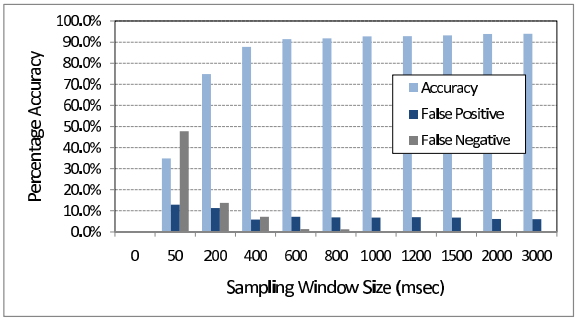
In our analysis we evaluate the performance of the *Long idle* technology. First we quantify its traffic classification accuracy and then we evaluate the power and performance impact in case of both idle and active workloads with and without using *Long idle*.

5.1 Traffic categorization accuracy

Long Idle relies on the NIC's ability to categorize the network traffic based on the conditions and algorithms outlined in Section 4. We use real-network traffic traces (as mentioned in Section 3) that represent hours of idle and active platform network activities as an input to our *Long idle* simulator running the classification and traffic-type-specific data handling algorithms to quantify the accuracy of these algorithms.



(a) Wired Environment



(b) Wireless Environment

Fig. 10. Long Idle Classification Algorithm Accuracy

Figure 10 analyzes the accuracy of the traffic classification algorithm in two environments wired home environment and wireless enterprise environment. The trend of this result is the same for all other tested networks. The false positive ratio represents the percentage of time idle traffic has been classified as active (i.e., the platform is waken up unnecessarily to process these idle packets) to the total time until it is correctly classified as idle. The false negative ratio represents the percentage of time active traffic is classified as idle (i.e., packets are held in the NIC and processing is delayed).

Figure 10 illustrates the effectiveness of our traffic categorization algorithm. As shown in the figure the average accuracy of the algorithm at a sampling window size of 1 sec in an enterprise environment is about 92% and in a home environment is close to 96%. The false negative percentage is very small which indicates that active traffic is never classified as idle, as a result active traffic is not buffered in the NIC and hence the active traffic performance and the user experience are unaffected. As shown in the figure most of the classification errors are

false positives (idle classified as active) which indicates that a small percentage of the power reduction opportunities have been missed and the platform is waken up unnecessarily.We confirm these simulations with implementation results presented in the next subsections.As expected the classification accuracy improves with increasing the sampling window but, on the other hand, the response time defined as how fast the platform will react to traffic changes degrades. As shown in the figure the classification accuracy saturates mostly at a sampling window size of 700 msec. Based on our results a sampling window size of 1 sec achieves a good classification accuracy and good responsiveness with unaffected user experience.

5.2 Idle traffic impact

We implemented *Long Idle* classification algorithm in a prototype Intel WiFi 5350 NIC *Intel WiFi Link 5100 Series Specifications* (2008). In our implementation we used a sampling window size of 1 sec and NIC idle traffic buffering timeout of 300 msec.

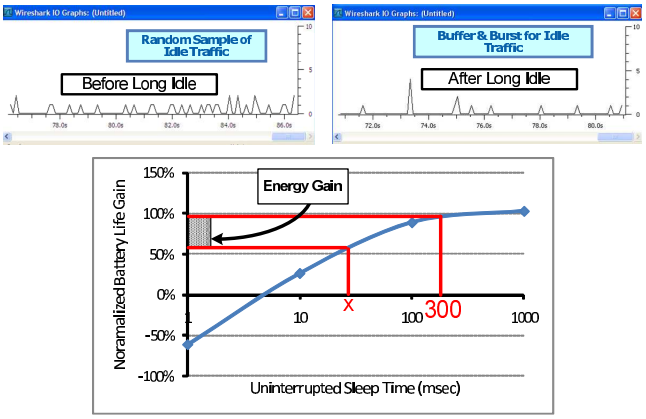


Fig. 11. Implementation Impact on Idle Traffic

Figure 11 plots the behavior of the NIC with idle traffic. Whenever the platform is not engaged in an active communication *Long Idle* technology will guarantee a quietness period as large as the buffering timeout value (e.g., 300 msec) set in the NIC and therefore the platform will stay in a low-power state uninterrupted during this time. Compared to Figure 5(b), energy gain achieved by Long Idle is about 23% for wireless environment and about 41% for the wired environment. It should be noted that we assumed that $\tau_{idle} = 0$ (see Section 3) to report worst case benefit and any value for $\tau_{idle} > 0$ will increase the energy gain of *Long Idle* even further.

5.3 Active traffic impact

To quantify the effect of *Long Idle* on active workloads we used our NIC prototype implementation to run various NetIQ Chariot *NetIQ Chariot 4.0 testing tools* (2001) benchmarks. NetIQ Chariot is a prepackaged real-world benchmarking tool widely used for network performance evaluation. We present results for FTP throughput and VoIP Mean Opinion Score (MOS) tests.

Table 1 represents the network throughput achieved by the NIC with and without *Long Idle* compared to the traditional 80.11 PSM. As shown in the table, the network throughout is

Input Rate	802.11 Observed Rate	Long Idle Observed Rate
256 Kbps	256 Kbps	256 Kbps
512 Kbps	512 Kbps	512 Kbps
1 Mbps	0.995 Mbps	0.985 Mbps
2 Mbps	1.997 Mbps	1.964 Mbps
10 Mbps	9.875 Mbps	9.7656 Mbps

Table 1. FTP Throughput with Long Idle

not degraded when *Long Idle* is used which indicates that the NIC correctly classified the FTP session as an active communication session and backed off from buffering any network packets. The slight difference in throughput between 802.11 PSM and that achieved by *Long Idle* is because *long Idle* has a slower responsiveness, as mentioned in Section 5.1, and the "first" packet in an active communication is delayed until at least one sampling window (i.e., 1 sec) has elapsed before the ongoing traffic is classified as active.

VoIP Session	802.11 PSM Quality	Long Idle Quality
Two Way	4.37 4.37	4.34 4.37
One Way	4.37	4.33

Table 2. VoIP Quality with Long Idle

VoIP MOS is a numerical method of expressing voice and video quality it is a measurable indication of the perceived quality of the media received after being transmitted and eventually compressed using codecs. A MOS value larger than 4.0 is what VOIP services targets as a good quality VoIP session *ITU-T Rec. G.729 Annex B, A silence compression scheme for G.729 optimized for terminals conforming to Rec. V.70* (1996). Table 2 represents the MOS results achieved by the NIC with and without *Long Idle*. Similarly, when *Long Idle* is enabled it correctly identifies the active communication and the packets are not buffered in NIC and hence no user experience impact. Similar to the FTP case, the small difference in the MOS score is attributed to the delay experienced by the first packet.

6. Conclusion

Typical platform power management relies on guiding individual platform components or the whole platform into low energy *(sleep)* states when the platform has been observed "idle" for some time with no active workloads. Individual power management techniques differ in how deep the sleep state is, the algorithms used to enter and exit the sleep states and optimizations to extend these sleep states as long as possible. In this chapter, we showed that significant energy is wasted at the platform level while it is idle and connected to a network because the system is processing background and management traffic. We showed that these background packets arrive at a rate high enough that can prevent the system from taking full advantage of the platform-level power management states. We quantified the negative impact of network connectivity on platform energy-efficiency and showed.

We proposed *Long Idle* a low overhead technique that is implemented inside the network interface card with no dependency from the host, application or the network infrastructure to classify the ongoing traffic into active versus background without deep packet inspection and to locally buffer the background traffic in the NIC to guarantee uninterrupted sleep periods for the platform. We showed that when our scheme is used the total platform sleep time increases by up to 40% with no performance degradation and no user experience impact.

7. References

Agarawal, Y., Hodges, S., Chandra, R., Scott, J., Bahl, P. & Gupta, R. (2009). Somniloquy: Augmenting network interfaces to reduce pc energy usage, *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

Agarwal, Y., Pering, T., Want, R. & Gupta, R. (2008). SwitchR: reducing system power consumption in multi-clients multi radio environment, *IEEE International Symposium on Wearable Computers (ISWC)*.

Flinn, J. & Satyanarayanan, M. (2004). Managing battery lifetime with energy aware adaptation, *ACM Transactions on Computer Systems*.

Gleixner, T. & Molnar, I. (2006). Dynamic ticks, http://lwn.net/Articles/202319/.

Gunaratne, C., Christensen, K. & Nordman, B. (2005). Managing energy consumption costs in desktop pcs and lan switching with proxying, split tcp connections and scaling of link speed, *International Journal of Network Management*.

Gupta, M. & Singh, S. (2003). Greening of the internet, *ACM Sigcomm*.

IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (1999). IEEE.

Intel WiFi Link 5100 Series Specifications (2008). www.intel.com.

ITU-T Rec. G.729 Annex B, A silence compression scheme for G.729 optimized for terminals conforming to Rec. V.70 (1996).

Li, X., Gupta, R., Adve, S. & Zhou, Y. (2007). Cross component energy management: Joint adaptation of processor and memory, *ACM Transactions on Architure and Code Optimization*.

Nedevschi, S., Chandrashekar, J., Liu, J., Nordman, B., , Ratnasamy, S. & Taft, N. (2009). Skilled in the art of being idle: Reducing energy waste in networked systems, *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

Nedevschi, S., Popa, L., Iannaccone, G., Ratnasamy, S. & Wetherall, D. (2008). Reducing network energy consumption via sleeping and rate adaptation, *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

NetIQ Chariot 4.0 testing tools (2001). http://www.netiq.com.

Olsen, C. & Narayanaswami, C. (2006). PowerNap: An efficient power management scheme for mobile devices, *IEEE Transactions on Mobile Computing*.

Shih, E., Bahl, P. & Sinclair, M. (2002). Wake on wireless: An event driven energy saving strategy for battery operated devices, *IEEE International Conference on Mobile Computing and Networking (MobiCom)*.

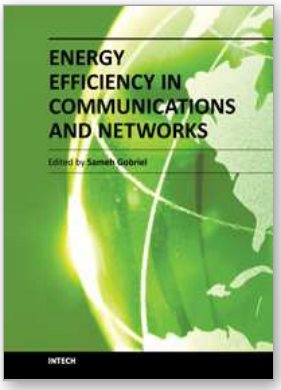
Siddha, S., Pallipadi, V. & Ven, A. V. D. (2007). Getting maximum mileage out of tickless, *Linux Symposium Proceedings*.

Sorber, J., Banerjee, N., Corner, M. & Rollins, S. (2005). Turducken: Hierarchical power management for mobile devices, *IEEE International Conference on Mobile Systems, Applications and Services (MobiSys)*.

TC38-TG4 Proxying support for sleep modes Specifications (2009). http://www.acpi.info/.

Windows Performance Analysis Tools (2008). http://msdn.microsoft.com/en-us/performance/.

Yodaiken, V. & Barabanov, M. (1997). A real-time linux, *Linux Journal*.



Energy Efficiency in Communications and Networks

Edited by Dr. Sameh Gobriel

ISBN 978-953-51-0482-7
Hard cover, 142 pages
Publisher InTech
Published online 04, April, 2012
Published in print edition April, 2012

The topic of "Energy Efficiency in Communications and Networks" attracts growing attention due to economical and environmental reasons. The amount of power consumed by information and communication technologies (ICT) is rapidly increasing, as well as the energy bill of service providers. According to a number of studies, ICT alone is responsible for a percentage which varies from 2% to 10% of the world power consumption. Thus, driving rising cost and sustainability concerns about the energy footprint of the IT infrastructure. Energy-efficiency is an aspect that until recently was only considered for battery driven devices. Today we see energy-efficiency becoming a pervasive issue that will need to be considered in all technology areas from device technology to systems management. This book is seeking to provide a compilation of novel research contributions on hardware design, architectures, protocols and algorithms that will improve the energy efficiency of communication devices and networks and lead to a more energy proportional technology infrastructure.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Sameh Gobriel, Christian Maciocco and Tsung-Yuan Charlie Tai (2012). Energy Efficiency of Connected Mobile Platforms in Presence of Background Traffic, Energy Efficiency in Communications and Networks, Dr. Sameh Gobriel (Ed.), ISBN: 978-953-51-0482-7, InTech, Available from: http://www.intechopen.com/books/energy-efficiency-in-communications-and-networks/energy-efficiency-of-connected-mobile

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821