Proceedings of the International Symposium on Sustainable Systems and Technologies, v3 (2015)

Is there a Net Energy Saving by the use of Task Offloading in Mobile Devices?

Vishnu Cherusola Dev Wichita State University, vxcherusoladev@wichita.edu
Vinod Namboodiri Wichita State University, vinod.namboodiri@wichita.edu
Milind Tandel Wichita State University, mhtandel@wichita.edu
Vijay Venkitachalam Wichita State University, vxvenkitachalam@wichita.edu

Abstract. The cloud computing paradigm enables the work anywhere anytime paradigm by allowing application execution and data storage on remote servers. This is especially useful for mobile computing and communication devices that are constrained in terms of computation power and storage. Relying more on powerful remote servers further allows each individual device's hardware to have reduced functionality, thus making them cheaper. Though there could be energy and performance benefits for mobile devices by offloading application tasks to remote servers, it is not clear whether there would be reduction in total end-to-end energy consumed when including energy consumed at the remote server and the network in between. The goal of this paper is to characterize the overall end-to-end energy consumed for task offloading in mobile devices. An analytical model of end-to-end energy consumed for task offloading is proposed and evaluated to understand the theoretical limits within which task offloading can save energy. Additionally, an empirical measurement-based evaluation is done with common off-the-shelf mobile devices to determine whether task offloading is more energy-efficient end-to-end compared to a scenario where the task is executed locally.

Introduction. Computing with mobile devices have always presented challenges in terms of storage, memory, processing, network connectivity, bandwidth, and battery lifetime in comparison to their static counterparts like desktop computers. With the technological advances in recent years improving ubiquitous connectivity and bandwidth, cloud computing has become feasible allowing these constrained devices to utilize the greater storage, memory, and processing capabilities of powerful remote servers.

Proceedings of the International Symposium on Sustainable Systems and Technologies (ISSN 2329-9169) is published annually by the Sustainable Conoscente Network. Jun-Ki Choi and Annick Anctil, co-editors 2015. ISSSTNetwork@gmail.com.

Cite as: Is there a Net Energy Saving by the use of Task Offloading in Mobile Devices? *Proc. ISSST*, Vishnu Cherusola Dev, Vinod Namboodiri, Milind Tandel, Vijay Venkitachalam. http://dx.doi.org/10.6084/m9.figshare.1536519 v3 (2015)

Cloud computing is typically a client server architecture, where the client can be any mobile device like a laptop, phone, browser, or any other operating system enabled device. Users of mobile devices that want to share documents, check email, and surf the Internet on the go, represent an increasing segment of the population that can utilize the cloud concept. Google's introduction of the cloud based Chrome operating system (OS), for example, fuels this trend of increasingly relying on remote servers instead of local device resources.

The energy implication of offloading a task or set of tasks (termed task offloading) to a remote server for execution is not very clear. On one hand relying on a server can reduce energy spent on task computation; on the other hand, communicating the task to the server will incur network-wide energy consumption. Such task offloading scenarios have been studied extensively for performance benefits or for exploring the feasibility of thin-client computing [1], [2], [3], [4], [5],[6], [7]. More recently such paradigms have also been studied in terms of reducing energy consumed by mobile devices and increasing battery lifetimes [8], [9], [10], [11], [12], [13]. These results have provided good guidance on when it is beneficial in terms of energy consumption for a mobile device to offload tasks. However it is not clear whether there will be benefits end-to-end when the energy consumption of the network and the remote server are brought into consideration. A broader study of the energy implications of task offloading from an end-to-end perspective is still missing. Such an end-to-end study would be one of the first steps in the path towards environmentally sustainable mobile computing. When it comes to looking at energy efficiency and the concept of sustainability in information and communication technologies (ICT), the focus has invariably been on data centers and mobile infrastructures like cell towers, as these have been considered the most power intensive within the computing sector. With an increasing emphasis on utilizing cloud computing through mobile devices, the study and analysis of the complete end-to-end scenario from mobile devices to end-servers is needed to further sustainability efforts in the area of information and communication technologies (ICT).

The specific novel contributions made by this work are the following:

- An analytical model of energy consumption for task offloading to remote servers is developed that includes the end-user device energy consumption, the remote server energy consumption, and the energy consumed by the network in between.
- An evaluation of the theoretical limits of various parameters in the analytical model is done for which task offloading can be shown to reduce overall end-to-end energy consumption
- An empirical evaluation of common-off-the-shelf mobile devices is performed to determine whether task offloading can reduce end-to-end energy consumption. Our results with a remote server of modest capabilities and a highly computational task shows that a end-user device like a laptop with only slightly lesser computational capabilities (than the remote server) could reduce end-to-end energy consumption by 30% easily with task offloading. For a much more relatively computationally-constrained device like a smartphone, the energy consumption can be considerably reduced.

Problem Statement. This section will focus on an analytical characterization of the task offloading problem with a formal problem definition, and an evaluation of the theoretical limits of model parameters under which task offloading can reduce overall end-to-end energy consumed.

Problem Definition. We will consider one mobile device (also termed as the client) that has to execute a task. It has the following two options: execute this task on the device locally, or offload the task to a remote

server over the network and have it executed there and have the results communicated back. Let E_{noff} be the total energy consumed to execute the task on the client device itself when no offloading is done. Thus, this energy consumption under the no offloading scenario to execute the task equals the product of the average power consumed to execute the task locally (P_c) and the time to taken the complete the task t_c.

$$E_{noff} = P_c * t_c \quad \ldots\ldots (1)$$

Under the offloading scenario, the energy consumed end-to-end is the sum of (i) energy consumed by the client E_{co}, (ii) energy consumed in the network E_{nw}, and (iii) energy consumed at the remote server executing the task E_{serv}. That is,

$$E_{off} = E_{co} + E_{nw} + E_{serv} \quad \ldots\ldots (2)$$

$$= P_{co} * t_{co} + P_{nw} * t_{nw} + P_s * t_s \quad \ldots\ldots (3)$$

For a given task, we can say that offloading saves energy end-to-end if

$$E_{off} < E_{noff} \quad \ldots\ldots (4)$$

Theoretical Limits. Here we numerically investigate the theoretical limits within which offloading could reduce the energy consumed for executing a task. We introduce some assumptions and some parameters to evaluate the condition in Equation 4.
Let $p_{nw/c}$ be the ratio of average power consumed by the network to that of the client to execute the task. That is $p_{nw/c} = P_{nw}/P_c$. Similarly, let $p_{s/c} = P_s/P_c$ be the ratio of average power consumed by the server to execute the task compared to the client locally. If we introduce another ratio $p_{co/c}$ to denote the ratio of average power consumed by the client when offloading versus executing the task locally, Equation 2 can be re-written as

$$E_{off} = P_{co*}\, t_{co} + p_{nw/c}/p_{co/c} * P_{co} * t_{nw} + p_{s/c}/p_{co/c} * P_{co} * t_s \quad \ldots\ldots (5)$$

Equation 1 can be re-written as

$$E_{noff} = P_{co}/\, p_{co/c} * t_s/r_{s/c} \quad \ldots\ldots (6)$$

where $r_{s/c} = t_s/t_c$ is the ratio of relative capabilities of the server and client respectively in terms of processing times for the same task. From Equations 5 and 6, the energy saving condition for task offloading reduces to

$$t_{co} + (\, p_{nw/c}/p_{co/c}\,) * t_{nw} + p_{s/c}/p_{co/c} < \; t_s / (p_{co/c} * r_{s/c}) \quad \ldots\ldots (7)$$

If we neglect the delays for packet processing, transmission, and queuing for simplicity, and the total time for task offloading at the client t_{co} would be equal to the sum of the total time spent in the network (both directions), t_{nw}, and the execution time at the server t_s, or $t_{co} = t_{nw} + t_s$. If we also introduce a ratio $r_{nw/c} = t_{nw}/t_c$ as the ratio of time spent to offload the task over the network to the client local task execution time, then Equation 7 can be re-written as the condition.

$$\frac{1 - \frac{p_{co}}{c} * r_{s/c} - r * p_{s/c}}{r_{nw/c}(p_{co/c} + p_{nw/c})} > 1 \quad \text{.....(8)}$$

Numerical Evaluation. Here we evaluate the condition in Equation 8 for a range of values for the power and processing capability ratios introduced above. The ratios $p_{co/c}$, $p_{nw/c}$, $r_{s/c}$, and $r_{nw/c}$ are varied to study the impact on the range of $p_{s/c}$ over which task offloading consumes less energy. For a given value of $r_{nw/c}$, it can be observed from the results in Figure 1 that a smaller client offload to no offload power ratio $p_{co/c}$ increases the range under which energy can be saved. Thus, efficiency of the low-power state when a client is waiting for a server to return a completed task is of great significance. It can also be seen that a lower network to client offload power ratio $p_{nw/c}$ helps increase the range over which offloading is superior. For small values of $r_{s/c}$, offloading is always preferable for all values of $p_{s/c}$. That is, when the server can execute the task much faster than the client, offloading always saves energy overall. For $r_{s/c}$ closer to 1, offloading is better only at lower values of $p_{nw/c}$, $p_{co/c}$, and $p_{s/c}$. Comparing across different values of $r_{nw/c}$ Figure 1 shows that as an offloaded task takes more time to be communicated back and forth from the server, the impact of increasing values of $p_{co/c}$ and $p_{nw/c}$ becomes more significant. That is, as a task incurs more communication delays, the range of power values of model parameters under which task offloading saves energy becomes narrower. The analytical model and its evaluations provide a means to study the impact of different parameters relative to each other and quantify their impact. The presented model is also useful to generalize the potential of end-to-end energy savings for any combination of end-user device, server, and network power consumption and processing capabilities, and for any type of tasks. In the next two sections, we look at specific current COTS devices and deployed networks and study whether offloading would reduce overall energy consumed for a specific task under consideration.

Empirical Methodology. In this section we describe our empirical approach to understand the conditions under which task offloading could save energy end-to-end. This section will focus more on the experimental methodology with evaluation results presented in the following section.

Overview. The empirical approach was divided into two cases. The first case involved measuring the energy consumed to execute a defined task only on a mobile device. The second case involved the mobile client offloading the task to a remote server which the latter completes and sends back. The energy consumed at the client, the server, and the network in between is measured in the second case with the sum of the three reported as the total energy consumed to execute the task. The results of the energy measurements from these two cases are then used to evaluate the condition first presented in Equation 4 if $E_{off} < E_{noff}$.

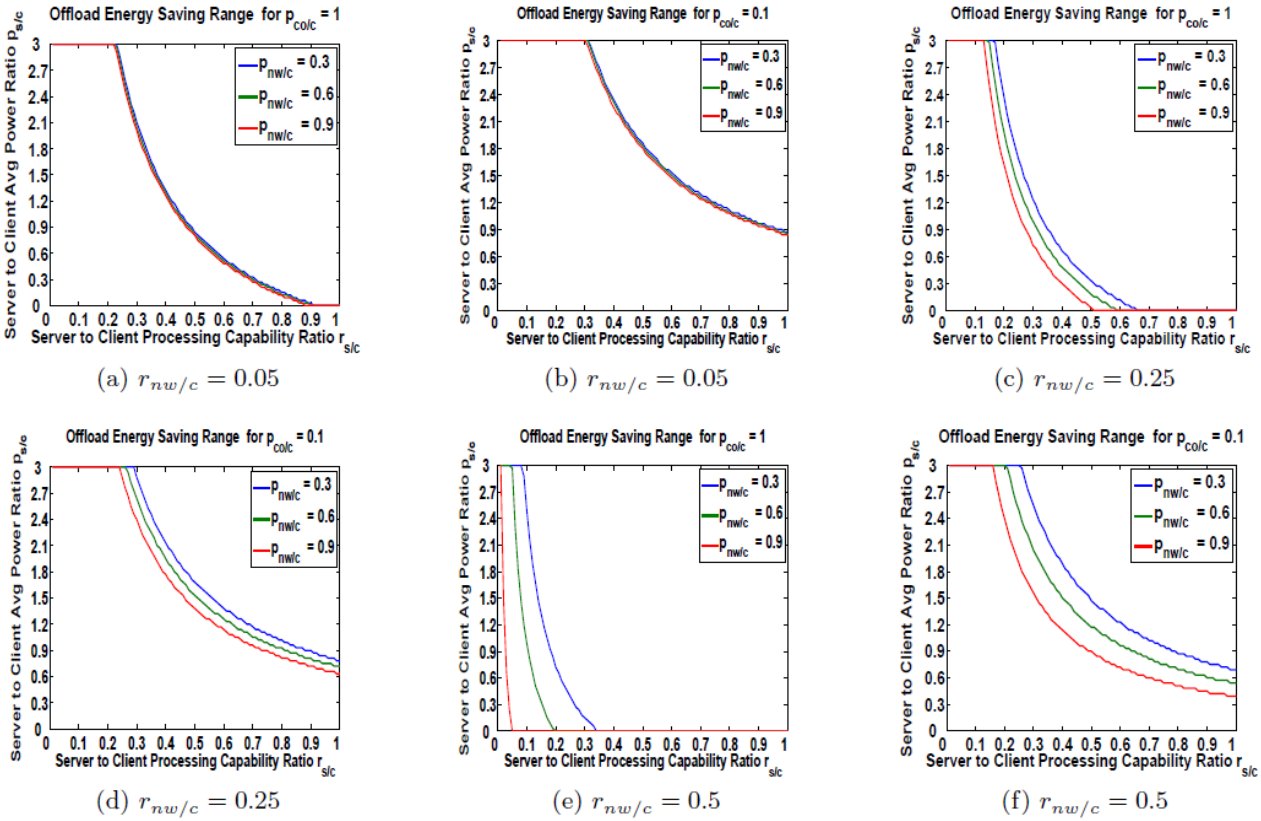


Figure 1. Impact of various parameter values on range over which task offloading can save energy over local execution with r_{nw}=c = 0.05, 0.25, and 0.5, and p_{co}=c = 0.1 and 1.

Experimental Setup.

Computing Devices. Mobile clients of two classes (based on computational capability) were tested, a laptop, and two smartphones. The laptop was a Lenovo SL3000 with an Intel core 2 duo CPU with each core running at 2.4GHz and 2 GB RAM. One of the smartphones was a Motorola Milestone XT 270 and ran Android OS v2.1 with 600 MHz CORTEX-A8 CPU. The other smartphone was a HTC Desire A8181 with a 1 GHz Scorpion CPU from Qualcomm and ran Android OS v2.2. All devices used the Wi-Fi interface (802.11g) for communicating over a network.1 The server machine emulating a cloud server was a more powerful laptop machine with an Intel core i3 CPU running at 2.53GHz and 4 GB RAM. Thus, the laptop end-user client was only slightly less computationally-capable than the server, while the smartphones were much more computationally-constrained than the server. Between the smartphones, the Motorola device considered was more computationally capable.

Task. The task considered was to sort a file containing *N* random integers repeatedly 200,000 times using the common insertion sort algorithm. Sorting is a task whose computation intensity and execution time can be easily varied by varying *N* making it very useful for the problem under consideration. Insertion sort is a commonly used sorting algorithm that is well-understood. Note that the particular sorting algorithm used could be easily changed and will not have any bearing on the conclusions drawn. The time to complete the task at the client (no offloading scenario)

involved using a simple timer. For the time to complete a task at the server (offloading scenario) the timer was set at the client before the task was sent over the network and the value was saved when the results were received back. Due to the wide variance in processing capabilities, we kept N = 10000 for the laptop and N = 100 for the smartphone. Experiments were also conducted to understand the impact of task complexity on the results by varying the size of the sorting input as N; $2N$; $4N$, and $8N$

Power Measurements. Energy consumption was computed as the product of average power consumed over the time to complete the task. The time duration to complete the task was explained above. The power consumption was measured through the Watts up? PRO meter for all devices except the smartphone for which the Monsoon Power Monitor tool was used. The Watts up? PRO meter allows recording power values over a period of time over a USB port and can provide average power and energy consumed as output. This meter had adequate resolution for capturing tasks that took well over 5-10 seconds to complete and has a stated accuracy range within ±1.5% of the actual value. For the smartphone, the commonly used Monsoon Power Monitor was used. It has a stated accuracy range within ±1% of the actual value.

Populating the Energy Model. Evaluating the energy consumed for the offloading and no offloading scenarios involves calculating each term in the energy models proposed in Equations 1 and 2. Energy consumed with no offloading: Calculating E_{noff} in Equation 1 was as simple as measuring the average power consumed by the client to complete the task locally Pc and multiplying it by the time taken to complete the task tc. Pc was computed by subtracting out a "base" power value for the device when idling and thus including only the additional power consumed to execute the task. This base power value denotes the power consumed by the device when it is not executing any tasks and includes power to run the device's display and the Wi-Fi interface in a low-power sleep state when not actively communicating.

Energy consumed for task offloading. Calculating E_{off} in Equation 2 was more involved due to multiple components spread over the network including a remote server. Our server as mentioned before was a more powerful laptop device. The network between the two was a LAN environment to get a best case result on top of which an analytical model was superimposed to vary the actual server location and WAN network characteristics.

For energy consumed by the client device when offloading t_{co} was computed by computing the time required to send the task to the remote server and getting it back, including the communication network latency. P_{co} was computed over this time and was found to be close to the base power of the device when it was awaiting results back from the remote server with spikes only for packet processing.

For energy consumed by the server device when executing the task, the energy consumed was found in a similar fashion to the client device. The time taken to complete the task at the server, t_s, was computed along with the average power consumed in the process P_s after subtracting off the base idling power of the server device. Note that $t_s < t_{co}$, with the difference primarily being the round trip network delay between the client and server. We also account for the overhead due to heating and cooling costs by the commonly used Power Usage Effectiveness (PUE) factor for only this server giving

$$E_{serv} = PUE * P_s . t_s \ldots \quad (9)$$

According to the Uptime Institute in a survey of data centers in 2012, the typical range of PUE was between 1.8 to 1.89. Thus we used the value of 1.8 for PUE in our calculations. Finally, we computed the energy consumed in the network. A router's energy consumption was modeled as in [15] by

$$E_{router} = V_r * P_r / C * U \ldots \quad (10)$$

where V_r is the total routed data volume in bits over the period of consideration, P_r is the average power consumption of the router, U is the utilization of the router, and C is the data handling capacity of router in bits per second. If we ignore protocol overhead, the data volume in our case will be equal to the data sent as part of task offloading to the remote server and the result back from the server. This is easily known based on the number of integers N that are being sorted for the example considered in this paper. We assume a utilization U of 25% as done in [15]. The parameters P_r and C can be easily found from specifications of a specific router.

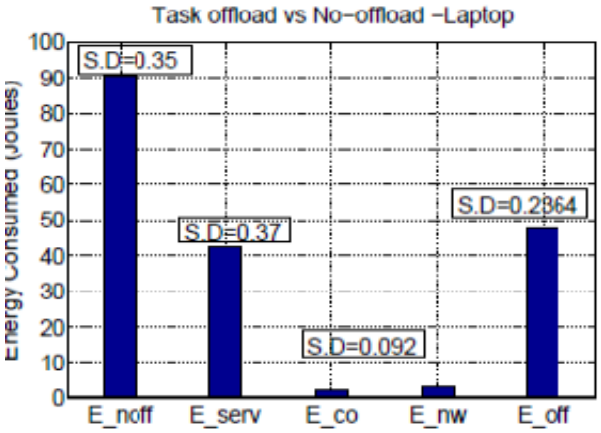
Our network model consisted of a client router, k_e edge routers, and k_c core routers. The overall energy consumed in the network was computed as

$$E_{nw} = E_{user} + k_e * E_{edge} + k_c * E_{core} \ldots \quad (11)$$

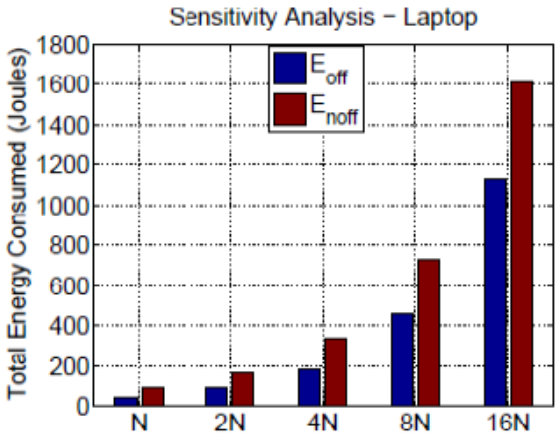
where we put in values corresponding to specifications for each type of router as listed in Table I. Our results in the next section are presented with k_e = 2 and k_c = 18 to model a long-distance WAN with 21 network hops overall. Note that this assumption likely overestimates the energy consumed by the network as, in most cases, k_c is likely to be much smaller than 18.

Empirical Results. In this section we present results empirically measured on different COTS mobile devices by applying the methodology described in the previous section. The first set of results are for a laptop, an end-user device which was only slightly less computationally-capable than the server, while the second set of results are for smartphones, a much more computationally constrained device than the server. The presented results try to answer two main questions for each class of end-user device: (i) how does the energy consumption compare between executing the task locally and remotely, and how does the energy consumption in the offloading scenario break down across the individual components E_{co}, E_{nw}, and E_{serv}, and (ii) how does the energy consumed in the offloading and no offloading case vary as the computation required (and hence processing duration) by the task increases. Each experiment was run 12 times with plots showing the mean value and standard deviation denoted S.D.

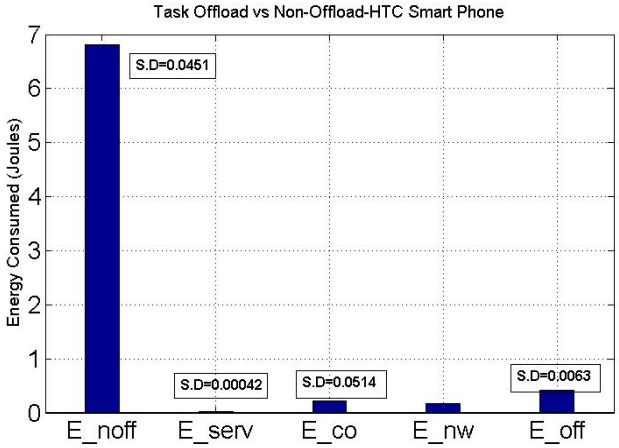
Results for laptops. The initial result of interest for laptops was whether task offloading would save energy in laptops, and by how much. Figure 2(a) plots E_{noff}, E_{off}, and the individual terms that comprise E_{off}, namely, E_{co}, E_{nw}, and E_{serv}. All terms except E_{nw} were measured empirically by repeatedly taking 12 different measurements with the mean value shown in the plot. The standard deviation of these measurements is also indicated above each bar. It can be observed that task offloading reduced the energy consumed by almost 50% over the no-offloading scenario.



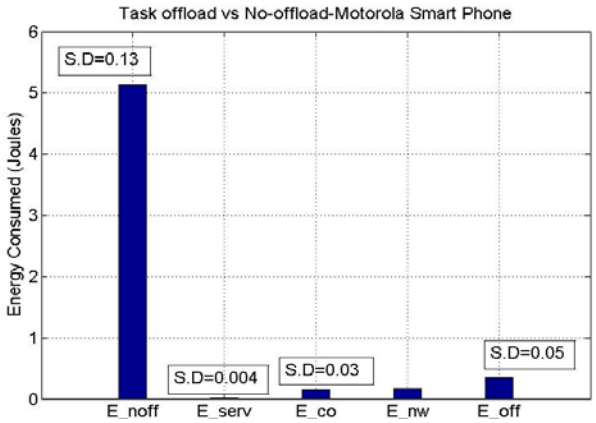
(a) N=10000 for Laptop, File size= 81.8KB



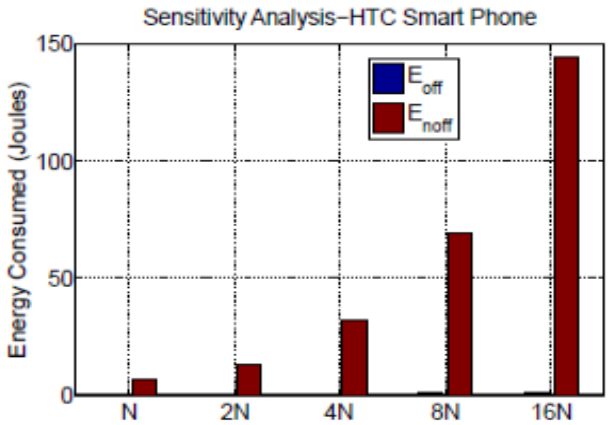
(b) Energy consumed for varying input size ona laptop



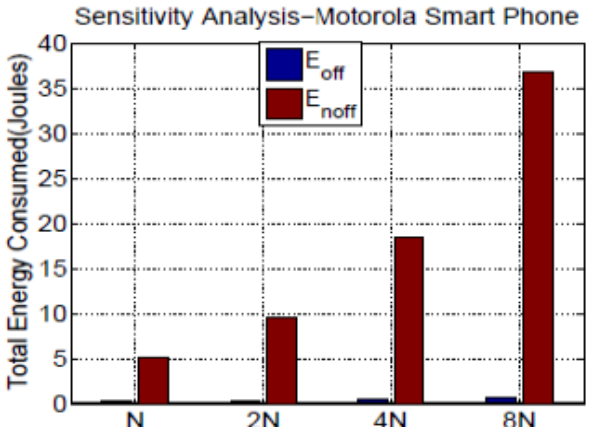
(c) N= 100 for HTC Smart Phone, File size=852 Bytes



(d) N=100 for Motorola Smart Phone, File Size=852 Bytes



(e) Energy consumed for varying input size on HTC Smart Phone



(f) Energy consumed for varying input size on Motorola Smart Phone

Figure 2. Energy consumed to execute task with and without offloading on laptops.

The energy consumed by the server to execute the task was the largest component in E_{off} with the network and client energy consumption relatively much smaller. The next result of interest for laptops was the impact of the complexity and duration to execute the task. This was studied by varying the number of integers to be sorted as per the sequence *N*, 2*N*, 4*N*, 8*N*, and 16*N* for

N = 10000. This study was more or less an analysis of the sensitivity of energy consumption to task complexity and duration. Figure 2(b) plots E_{noff} and E_{off} for varying input sizes. It can be seen that both increase with the increase in input sizes in line with the $O(n^2)$ complexity of insertion-sort. In absolute terms, greater the input size, greater the energy saved by offloading for the scenario considered. In relative percentage terms, however, the converse seems true with the ratio of *Eoff/ Enoff* decreasing with greater input size. For the inputs tested, and relative performance difference between the laptop and server used, 30% savings in end-to-end energy savings was easily possible by task offloading.

Results for Smartphones. The initial result of interest for mobile phones was again whether task offloading would save energy in a computationally-constrained form-factor device like smartphones and by how much. Plots for two different smartphones that we considered are shown in Figures 2(c) and 2(d). An important distinction from the experiments for laptops is that N is set now equal to only 100 due to the much lesser computational capabilities of these devices. Both plots show that even for smartphone, task offloading seems to consume much less energy, an even more pronounced difference than in laptops. This is because E_{noff} values are much larger for local execution in phones whose processing capabilities are limited and the duration to complete the task being relatively much larger than on the powerful server. This trend is seen on both phones, though E_{noff} is much smaller on the Motorola phone as it is newer hardware model with higher processing capability and a newer OS. The end-to-end energy consumption was reduced considerably in both cases. The next result of interest for smartphones was based on the sensitivity to input size. Just as the case of laptops, the input size was varied in multiples of two. Figures 2(e) and 2(f) show these results. The sensitivity analysis for both phones 2 to varying input size demonstrated that a powerful server can handle much larger input sizes easily compared to a phone and hence offers a continuing advantage that is much higher than what was seen for laptops.

Discussion of Results. The results for both classes of mobile devices studied in this section highlight that the benefits of task offloading increases as the computational-capability of the end-user device decreases. For the laptop, reduction in end-to-end energy consumption was only of the order of 30% because the server itself was a laptop, albeit a more powerful one. If a more commercial grade server were used, much greater benefits to task offloading could perhaps be seen. Surprisingly, the energy consumed in the network was much lesser than we had anticipated before our experiments in spite of the worst case assumption of 21 hops as network diameter. After studying these empirical results, it becomes apparent that the analytical model presented in Section II is a useful complementary contribution as well in generalizing when task offloading is likely to reduce end-to-end energy consumption. For example, one can compute (omitted due to space limitations) the range at which offloading will save energy for some given power consumption values and processing capabilities at the client, network, and server and predict the benefits of offloading for other unknown values.

Conclusions. This paper characterized the overall end-to-end energy consumed for task offloading in mobile devices. An analytical model of end-to-end energy consumed for task offloading was proposed and evaluated to understand the theoretical limits within which task offloading can save energy. Additionally, an empirical measurement-based evaluation was done with common-off-the-shelf mobile devices to determine whether task offloading is more energy-efficient end-to-end compared to a scenario where the task is executed locally. Our results show that task offloading can significantly reduce end-to-end energy consumed, with greater benefits seen when end-user mobile devices are less powerful; for a smartphone considerable reduction in energy

consumption is seen as opposed to a laptop for which energy reduction is 30%. Future work to be done could include studying how results may vary when the profile of tasks to be offloaded is changed. Multiple types of servers could also be experimented with studying the impact of server processing capability empirically. The analytical model of the network energy consumption could also be improved by adding more details.

References

[1] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, ser. ICDCS '02, 2002, pp. 217–226.

[2] N. Tolia, D. G. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *IEEE Computer*, vol. 39, no. 3, pp. 46–52, 2006.

[3] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, ser. MCS '12, 2012, pp. 29–36.

[4] S. Wang and S. Dey, "Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, no. 1, pp. 10–21, Jul. 2012.

[5] Y. Guo, L. Zhang, J. Kong, J. Sun, T. Feng, and X. Chen, "Jupiter: transparent augmentation of smartphone capabilities through cloud computing," in *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, ser. MobiHeld '11, 2011, pp. 2:1–2:6.

[6] S. Imai and C. A. Varela, "Light-weight adaptive task offloading from smartphones to nearby computational resources," in *Proceedings of the 2011 ACM Symposium on Research in Applied Computation*, ser. RACS '11, 2011, pp. 146–152.

[7] E. Miluzzo, R. Caceres, and Y. farn Chen, "mClouds: Computing on Clouds of Mobile Devices," International Workshop on Mobile Cloud Computing and Services (MCS '12) with MobiSys 2012, Tech. Rep., June 2012.

[8] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, ser. CASES '01, 2001, pp. 238–246.

[9] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and R. Chandramouli, "Studying energy tradeoffs in offloading computation/ compilation in java-enabled mobile devices," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, pp. 795–809, September 2004.

[10] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer*, vol. 43, no. 4, pp. 51–56, 2010. [11] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *MobiSys*, 2010, pp. 49–62.

[11] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11, 2011, pp. 301–314.

[12] V. Namboodiri and T. Ghose, “To cloud or not to cloud: A mobile device perspective on energy consumption of applications,” in *WOWMOM*, 2012, pp. 1–9.

[13] Cisco, “Product support and documentation,”http://www.cisco.com/cisco/web/support/index.html.

[14] D. Schien, C. Preist, M. Yearworth, and P. Shabajee, “Impact of location on the energy footprint of digital media,” in *Sustainable Systems and Technology (ISSST), 2012 IEEE International Symposium on*, may 2012, pp. 1 –6.

[15] M. Tandel and V. Venkitachalam, “Cloud computing in smartphone: Is offloading a better-bet?” Electrical Engineering and Computer Science, Wichita State University, Tech. Rep. TR-EECS-WSU-2012-002, 2012.