Offloading for Mobile Device Performance Improvement

Dagnachew Temesgene
CTTC
dagnachew.temesgene@cttc.cat

Jari Porras
Lappeenranta Univ. Of Tech.
jari.porras@lut.fi

Janne Parkkila
3BIT
janne@3bit.fi

ABSTRACT

Mobile devices are increasingly becoming part of everyday life. These include smart phones, tablets, wearable devices etc. Due to their mobility aspect, they are always constrained in their size and weight, which limits their resource capacity, e.g. processing power, and battery life. One possible solution for augmentation of such resource-constrained devices is through efficient usage of their surrounding resources, i.e. using some offloading technique. This paper studies how offloading of tasks to the surrounding resources affects on both the performance of task execution as well as the battery life of the mobile device. Two mobile phones and two tablets (from two different manufacturers) are studied in the experiments to find out the impact of the device characteristics. Two computationally demanding tasks, namely image processing and encryption/decryption, are used in these experiments. These results are compared to our earlier results on mobile devices of previous generations. We assumed that the increased computing power of new devices would make offloading obsolete. Our results show gains both in energy saving and in computational performance with these mobile devices. The comparison to our earlier results show that the performance increase of newer mobile device generations has not diminished the benefits of offloading. These results are in line with results presented in literature and they show that the offloading could offer a viable approach for resource augmentation of mobile devices towards edge/fog resources emphasized by the new 5G technology.

1 Introduction

Mobile devices are increasingly becoming part of everyday life. These include smart phones, tablets, wearable devices and sensors. However due to their mobility aspect, they are always constrained in their size and weight which limits their physical capabilities, especially the battery life. One possible solution for augmentation of such resource-constrained devices is through efficient use of other computing resources. The environment in which these devices exist has a lot of unused computing resources. Resource-constrained in this case means the limitation due to battery capacity and also other computing resources such as processor speed and memory. Though mobile devices are becoming more powerful, there still exists a gap between a performance of mobile devices and other computing machines such as desktop computers, laptops or even cloud-based computing centers (i.e. mobile devices will remain resource-contrained). Therefore, efficient utilization of these external computing resources could provide a viable solution to extend the capabilities of the mobile devices.

One approach to offloading computations is cyber foraging, meaning the opportunistic use of nearby computing resources to enhance the performance of mobile device [1]. It is originally construed as “living off the land,” with an idea to dynamically augment the computing resources of mobile devices by exploiting the computational resources of wired hardware infrastructure [1]. According to this original definition, the distinct feature of cyber foraging over other offloading or remote execution mechanisms is being an opportunistic offloading method. The recently appeared techniques, like edge and fog computing as well as a popular method cloudlets, are more structured and multi-tier approaches than cyber foraging that aims at direct connection between the mobile device and external resource.

It is important to note that in parallel with the increasing computing capacity of mobile devices, the complexity of computing tasks that the mobile devices need to support is also growing. The users expect their mobile devices to perform an increasing amount of computations. More resource demanding applications such as augmented reality applications are expected to appear in mobile devices. These more resource demanding applications will then challenge the limits of mobile devices. Although the performance of mobile devices is increasing the question lies if our needs for various applications are increasing faster and as such the performance gap would widen. In this paper we wanted to study how the use of offloading computationally intensive tasks would affect the performance of a mobile device. We did similar study in 2010-2011 and were now interested if the improved capabilities of new mobile device generations would have diminished the benefits of offloading. Our assumption was that the improved computational performance of new mobile devices has made offloading almost obsolete. Due to the earlier study [14] we used the same Scavenger cyber foraging system although there are newer tools available. Scavenger allows efficient offloading from one device to another as long as there is connection between these places and this simple operation if enough for this offloading research. We use cyber foraging system scavenger to evaluate the aggregate energy saving, performance improvement and battery saving advantages of scenarios in various operation modes (like opportunistic and controlled).

The rest of the paper is organized as follows. Section 2 describes the past research works that are related with this paper. A more detailed view of cyber foraging system used in this study is given in Section 3. Section 4 explains the research process used in the research. The results are given in section 5 and we draw our conclusions in section 6.

2 Related work

Various techniques of offloading computation to a more resource rich devices are being used today. One of these techniques is cloud computing which is a paradigm for hosting and delivering computing services over the Internet. The primary goal of cloud computing is to maximize the usage of various computing resources in order to overcome the need of resources for high performance computation and to achieve a higher throughput [3][4]. Its main features include elasticity, resource pooling, on-demand self-service and broad network access [5]. Various deployment models are being used which includes public clouds, private clouds and hybrid clouds. Cloud computing is a major paradigm shift in computing technology with its approach of providing computational resources as per usage similar to utilities such as electricity. This is possible due to the increasing availability of high-speed wired and wireless networks. Another related technique to consider is mobile cloud computing. It is a paradigm at the forefront of mobile and cloud computing and is defined as a technique for integration of Cloud Computing technology in mobile environment and provides all the necessary resources to overcome the obstacles of the mobile devices [6][7]. Several definitions exist for mobile cloud computing. For instance, mobile cloud computing is defined as running mobile intended applications such as email and social networking apps on remote resource rich server [8]. The same author also considers mobile devices as resource providers in making of mobile cloud through peer-to-peer network. Another concept proposed by [9] is an approach of using small-scale servers called cloudlets at the edge of Internet. These small-scale servers can be deployed in public places such as coffee shops and supermarkets where the mobile client can make use of their storage and processing resources. They are connected to a cloud data center by high speed wired connections. The obvious advantage of these approaches is a reduction of latency in applications since mobile device requests need not go far to a remote data center but rather served by the close cloudlet [9]. A number of use cases and opportunities are possible by mobile cloud computing. In [10] and [8], the use cases are categorized into image and natural language processing, sharing GPS and Internet access, applications using sensor data, querying, crowd computing and multimedia search. A similar concept called mobile edge computing is also having more attention recently. It is a trend, which aims to solve the problem due to large volumes of data traffic from mobile users and associated latency as well as resource and energy limitations of mobile devices. It is defined as a cloud server running at the edge of a mobile network and performing specific tasks on behalf of mobile clients that could not be achieved with traditional network infrastructure [11]. The concept of edge computing is similar to mobile clouds with the aim of augmenting mobile devices by resource rich servers deployed at the edge of a mobile radio network. Such edge servers can be owned by mobile network providers and the subscribed mobile devices can use them by offloading part of their computation (this helps to minimize latency). On the other hand, even though the original definition of cyber foraging implies opportunistic offloading, some extend this definition to include offloading to a cloud as well [12].

This original definition of cyber foraging is extended in most cases to include the offloading of computation to a more resource rich dedicated server in mobile environment. Hence systems and approaches in mobile cloud computing and mobile edge computing can also be regarded as cyber foraging systems.

Our previous work on analyzing the battery gains (i.e. battery power savings) of offloading [13] shows that cyber foraging is advantageous for saving the battery life and improving runtime of intensive tasks. However, our work work also shows that offloading is disadvantageous both in battery life and performance for some tasks involving a large data transfer to and from a surrogate computer. These outcomes of our previous study set our initial assumption that the increased processing power of new mobile devices would in the end make the offloading obsolete as computing will be (or will become in near future) more efficient than communication. Another interesting fact in the previous study was that the compared devices (Nokia vs. Samsung) varied a lot in computing capabilities. We assumed that these differences would eventually diminish with new generations of mobile devices.

3 Cyber foraging

For a mobile device to take advantage of offloading (cyber foraging) the following six sub-processes should be considered [14]. 1) *surrogate discovery*: Mobile client applications need to discover the availability of surrogates that are capable of providing the necessary resources. This discovery is based on the use of available communication technologies. However, being discoverable does not mean that the surrogate is willing to share its resource. 2) *Application partitioning*: At this level the task or the application needs to be partitioned into locally executable and remotely executable parts: the task that requires more computing resources such as storage, processing, and communication speed is a good candidate for remote execution. 3) *Cost assessment*: The process of cost assessment defines where the application has to be executed. 4) *Trust and security establishment*: When a client's task is remotely executed on a surrogate, the task code needs to be protected from a malicious surrogate and the surrogate needs to be protected from a malicious client's tasks. 5) *Actual task execution:* The tasks that are suitable for remote execution are executed remotely using a surrogate computer. 6) *Environment monitoring*: This is necessary since the environment is dynamic and mobile hence the device needs to monitor the computing environment for any changes in the surrogates and in cases where a better option is available.

One cyber foraging system that is used for offloading is Scavenger. It was developed by Aarhus University, Denmark. The Scavenger system is based on Python and enables cross-platform support, mobility and low application development times [15]. This system focuses on computational offloading and consists of two software components. The first one is a Daemon, which runs on the surrogate computer and enables them to receive requests from clients and execute tasks. The second component is a client library for enabling client applications. Client application must use this library to enable scavenger applications. This library offers two ways of working with cyber foraging: a manual mode, where the application may itself ask for a list of available surrogates, install code onto these surrogates, and invoke this code; and a fully automated mode, where the above routines in manual mode operation are taken care of by scavenger scheduling [15]. Although Scavenger is not the newest offloading tool it provides the necessary functionality of running tasks locally and remotely and measuring the performance. In this study we used Scavenger to run our tests and emphasized the actual task processing and evaluation of it. Although the other sub-processes can make a big difference in actual performance of the task execution, we wanted to focus on the device dependent part and see if the evolution of mobile devices has changed the benefits.

4 Research process

This research is based on a set of measurements on a set of predefined set of use cases and experiments with predefined devices. Our assumptions for this research work are as follows.

- The growth in the power of mobile devices is balanced by the growth in the resources of potential surrogate devices and the increasing complexity of mobile computing tasks. This, together with the improvements in speed and latency of wireless networks, enables tasks that are processing intensive (but relatively small size of transferred data) to be offloaded. If compared to the previous measurements we expect that the benefits have been decreased due to smaller improvements in networking technologies and smaller increase in the complexity of computational tasks. However, we are eager to see how the new generations of mobile devices have managed to balance the computations and communications.
- To the best of our knowledge, there are no available works that are done to analyze the holistic energy savings of cyber foraging. The works that are done so far, analyze the gains of cyber foraging only from the resource constrained or battery powered benefit side. This research will try to test the aggregate sustainability advantage of cyber foraging.

Table 1 Mobile devices used and specifications. * devices from the previous study [13].

Device model	Battery	Processor	Memory	Screen size
Nokia N900*	Li-Ion 1320 mAh	ARM Cortex A8 600 MHz	256MB	3.5 inches
Samsung Galaxy Tab 7*	Li-Po 4000 mAh	ARM Cortex A8 1GHz	512MB	7 inches
Samsung Galaxy J5	Removable Li-Ion 2600 mAh	Quad-core 1.2 GHz Cortex-A53	1.5GB	5.0 inches
Samsung Galaxy Tab A 9.7	Non-removable Li-Ion 6200 mAh	Quad-core 1.2 GHz	1.5 GB	9.7 inches
Huawei MediaPad M2	Non-removable Li-Po 4800 mAh	Quad-core 2.0 GHz Cortex A53 + quad-core 1.5 GHz Cortex-A53	2GB	8 inches
Huawei Honor 6	Non-removable Li-Po 3100 mAh (11.8 Wh)	Quad-core 1.7 GHz Cortex-A15 & quad-core 1.3 GHz Cortex-A7	3GB	5 inches

The experiments are applied on chosen android mobile devices. These android devices and their specifications are given in Table 1. Note the two first devices are from our earlier experiments and we only compare the base results of these devices to the new set of devices. The surrogate characteristics for this new study are given in Table 2. The experimental configurations that are common for all of the tests are given in Table 3. In order to understand the behavior of various devices a set of baseline tests were run before actual offloading experiments.

Table 2 Characteristics

Parameter	Specification
Processor	Intel Core i3-3217U CPU @ 1.80GHz × 4
Memory	5.7 GiB
OS	Ubuntu 14.04 LTS
Battery	4Cells 2600 mAh 37 Whrs

Table 3 Experimental configurations

Property	Value
Screen brightness	100%
Network configuration	IEEE 802.11g, all connected to the same network
Surrogate cores used	1
Battery charge level prior to each test	100% (fully charged)

4.1 Baseline tests

The aim of these series of tests is to understand the battery characteristics of mobile devices that are used. Hence tests involving measurement of battery consumption in five modes namely i) idle mode, ii) idle mode with Wi-Fi on (pinging the network), iii) cpu loaded by computation intensive tasks, iv) cpu loaded and Wi-Fi on, and v) continuous download. These five test modes are all applied on each android device and corresponding measurements are recorded for one hour. As it is pointed in the experimental configuration, all of the devices are fully charged prior to applying each test case.

4.2 Offloading tests

In this set of tests, various tests with the aim of evaluating opportunistic offloading technique and competitive offloading scenarios are done. Two separate tasks with different resource requirements are executed i) locally, ii) locally with Wi-Fi on (worst case scenario) and iii) remotely with offloading. The tasks are simple encryption/decryption of text file and image edge detection. A more detailed explanation of the nature of these tasks and the motivations for selecting them is given in sub-section 2 of section 5.

The local execution is optimized so that no offloading overheads exist. Also in the case of local execution the Wi-Fi interface is kept off. In the case of local execution with Wi-Fi, the application is set to search for available potential surrogate resources in its vicinity but in the end the application ends up executing the task by itself due to the unavailability of remote resource. It is a worst case that can be faced by a mobile device (putting effort to find surrogate but in the end executing the task by itself). The third case is to execute the task remotely on a nearby surrogate device, which is willing to provide its resource to be used by a mobile device client. Measurement of battery consumption is done by using scripts that log an instantaneous battery level of android device. Offloading tests will consider two scenarios, namely:

- **Opportunistic offloading:** The aim of this scenario is to evaluate the results when a single mobile device tries to use the resources of a surrogate.
- **Competitive offloading:** In this scenario, offloading happens from multiple different devices simultaneously. Four devices are trying to access the resources of a single surrogate at the same time. This represents a mobile edge computing. Mobile devices use a close by dedicated server to improve their performance and to reduce battery energy consumption. In each scenario and task, the results are analyzed to evaluate the battery saving, performance improvement and aggregate energy saving advantages of offloading.

5 RESULTS

The results are presented first for the baseline tests and then for the offloading tests. Baseline tests of the four newer devices are compared with the baseline tests of our earlier study.

5.1 Baseline test results

Baseline tests are done for the five given variations and the tests were run for one hour each. The results of the baseline tests are shown in Appendix 1 in Fig 1 and Fig 2. The Y-axis of each device is different due to different battery capacities (some devices have larger batteries than others). However, each figure shows in a similar manner how the battery power is drained in each baseline test. In each device the idle mode (i) uses the least amount of energy as expected. Other modes show some differences.

The new set of modern devices show rather consistent behavior with idle mode with pinging (ii) being the second best energy-wise followed by networking (iii), computation (iv) and finally computation with pinging (v). CPU intensive tasks tend to drain more battery power than other tasks. This has changed when compared to our earlier results as in those results the early Samsung Galaxy tab device had more efficient downloading than pinging function and in Nokia N900 networking was much more inefficient. In the new set of devices it can be observed that tablets drain about 10% of battery charge while smartphones drain about 4.5% relative to their maximum capacity in idle cases for the duration of an hour. In addition, Huawei devices used in these tests drain more energy for CPU intensive tasks (a battery drain of 16% – 20% compared to 9 – 13% battery drain by their Samsung counterparts).

These results are taken as a basis for the analysis of offloading performance. The evaluation that CPU intensive tasks drain more energy can be an initial point to make an assumption that CPU intensive tasks are good candidates for offloading.

5.2 Offloading test results

Results of task 1, scenario 1: Encryption and decryption of a file in opportunistic offloading

This task involves a simple encryption and decryption of a text file of about 163Kb size. This section shows its execution in opportunistic offloading mode. This task is very good for offloading as both task code and data can be easily (low communication versus computation needs) migrated to a surrogate. These results are shown in Table 4. The results are divided into battery saving, performance improvement and total energy saving. The results show that opportunistic offloading provides a significant advantage for battery saving, performance improvement and aggregate energy saving. The amount of battery power consumed for the task via scavenger is only 3.3% to 12.69 % of the battery consumption of doing the task locally. The runtimes by offloading are only 9.47% to 18.74% of the runtime of doing the task locally and a significant energy saving is obtained in aggregate energy consumption of task execution. Though it also depends on the particular surrogate used, for this case, from 56% to 80% total energy savings are measured relative to the energy consumed by doing a task by the mobile device itself. Not only is opportunistic offloading advantageous in saving battery life but also it is efficient in terms of discharging rate as it is shown in mAh/sec result column. The mAh/sec result from offloading is lower than that of doing the task locally.

The overhead due to offloading when a desired surrogate is not available is very low compared to the advantage obtained by scavenging and offloading. The overheads range from 1.49% to 15.38% of the battery consumption of doing a task locally without searching for potential surrogate. The overhead of scavenging for available surrogate and doing the task locally has no effect in the runtime with only less than a percent increment relative to local task execution without searching for surrogate. The only exception is Huawei Honor 6 where 19% runtime increment is observed as overhead of searching for available surrogate. An interesting point to observe is that, though the devices have different performance in executing a task by themselves, their performance is enhanced to the same level by offloading.

Table 4. Opportunistic offloading results of encrypt/decrypt task (in comparison % negative numbers mean that the option is worse than the one it is compared to, positive numbers indicate better performance)

Device	Test type	Battery consumption in mAh	Discharge rate mAh/sec	mAh saving in % relative to local	Average runtime in seconds	runtime improvement in %	Total Energy (Wh)	Energy saving %
Huawei Mediapad	Local	67	0.2179		307.4		0.266	
	Local+wifi	68	0.2197	-1.49	309.43	-0.66	0.26955	
	Scavenge	8.5	0.1475	87.31	57.61	81.26	0.105766	60.24%
Huawei Honor 6	Local	91	0.159		572.33		0.3463	
	Local+wifi	105	0.154	-15.38	681.86	-19.14	0.3996	
	Scavenge	3	0.05556	96.70	54.4	90.49	0.083492	75.89%
Samsung Galaxy Tab A	Local	150.2	0.2534		592.6		0.5849	
	Local+wifi	155	0.2631	-3.19	589	0.61	0.6036	
	Scavenge	10.5	0.187	93.01	56.14	90.53	0.11295	80.69%
Samsung Galaxy J5	Local	47.8	0.0826		578.65		0.1816	
	Local+wifi	51	0.08738	-6.70	583.66	-0.86	0.1938	
	Scavenge	2	0.03485	95.82	57.39	90.08	0.079672	56.12%

Table 5. Competitive offloading results of encrypt/decrypt task

Device	Test type	Battery consumed (mAh)	Discharge rate mAh/sec	mAh saving %	Average runtime (secs)	runtime improvement in %	Total Energy by local execution	Total Energy by scavenging [Mobile side + Surrogate side] (wh)	Energy saving %
Huawei Mediapad	Local	67	0.2179		307.4		1.3788	0.568	58.84%
	Local+wifi	68	0.2197	-1.49	309.43	-0.66			
	Scavenge	28	0.1475	58.21	134.27	56.32			
Huawei Honor 6	Local	91	0.159		572.33				
	Local+wifi	105	0.154	-15.38	681.86	-19.14			
	Scavenge	12.5	0.05556	86.27	145.81	74.53			
Samsung Galaxy Tab A	Local	150.2	0.2534		592.6				
	Local+wifi	155	0.2631	-3.19	589	0.61			
	Scavenge	26.25	0.187	82.52	143.56	75.78			
Samsung Galaxy J5	Local	47.8	0.0826		578.65				
	Local+wifi	51	0.08738	-6.7	583.66	-0.86			
	Scavenge	4.5	0.03485	90.59	144.93	74.96			

Results of task 1, scenario 2: Encryption and decryption of a file in competitive offloading

In this part, results and evaluation of offloading computation for encryption and decryption (task 1) in the competitive offloading scenario i.e. when the four devices request the surrogate at the same time, are done. This scenario is for cases when multiple devices try to use the same available resources at the same time. These results are shown in table 5. The results show that competitive offloading scenario provides a significant advantage for battery saving and performance improvement as compared to doing the task locally. The amount of battery charge consumed for task via scavenger is only 9 - 42 % of the battery consumption of doing the task locally and the runtime are only 24% to 44% of the runtime of doing the task locally. However, these results are considerably lower than the savings obtained when the surrogate resource is used by a client alone. An aggregate energy saving of 58% is also obtained via competitive offloading scenario. The offloading results also show that it is efficient in terms of discharging rate as it is shown in mAh/sec result column. In addition, competitive offloading enhances the performance of end devices to the same level.

Results of task 2, scenario 1: Edge Detection of images in opportunistic offloading

In this task, 50 edge detection operations are performed on a 13 Mpix image and the test is repeated 5 times. This is done to approach offloading with a task that involves a significant data/file transfer to and from a surrogate. The 50 edge detection operations performed involve traffic of almost 300Mb to and from a surrogate.

Similar to the task 1, scenario 1 involves an opportunistic offloading towards a surrogate computer. These results are shown in Table 6. In this computing task, as it is the case for task 1, offloading provides a significant advantage. The battery charge consumed for task via scavenger is only 12% - 20 % of the battery consumption of doing the task locally and the runtime by offloading are only 15% to 25% of the runtime of doing the task locally. Not only is offloading advantageous in saving battery life but also it is efficient in terms of discharging rate as it is shown in mAh/sec result column.

The overhead due to offloading when a desired surrogate is not available is very low compared with the advantage obtained by offloading. The overheads range from 0.1% to 3.3% of the battery consumption of doing a task locally without searching for potential surrogate and only less than a percent increment in runtime relative to local task execution without searching for surrogate. The only exception is Huawei Honor 6 where 6.3% runtime increment is observed as overhead of searching for available surrogate. Also an energy saving of 23% - 70% is obtained by doing the task via offloading. Even though these savings are lower than the savings for task 1, it also shows the potential of offloading to save energy even for network intensive tasks. An interesting remark to note is that even though offloading results in a significant battery saving and performance improvement, these saving are lower than the case for task 1. This is attributed to the network consumption in sending and receiving images from surrogates. In addition, it is observed that, as in the case for task 1, all of the devices have almost similar performance by scavenger as it is shown in almost equal runtime by scavenging.

Table 6. Opportunistic offloading results of edge detection task.

Device	Test type	Average Battery consumption in mAh	Discharge rate in mAh/sec	mAh saving in % relative to local	Average runtime in seconds	runtime improvement in %	Total Energy (wh)	Energy saving %
Huawei Mediapad	Local	255	0.2114		1206		1.011	
	Local+wifi	258.4	0.214	-1.33	1207.52	-0.12	1.0243	
	Scavenge	50.6	0.1652	80.16	306.27	74.64	0.573	43.32%
Huawei Honor 6	Local	257.4	0.1961		1312.65		0.98	
	Local+wifi	249.4	0.1787	3.10	1395.4	-6.30	0.948	
	Scavenge	35.8	0.113	86.10	317	75.85	0.5082	48.14%
Samsung Galaxy Tab A	Local	542	0.254		2134		2.11	
	Local+wifi	543	0.2561	-0.18	2120	0.66	2.114	
	Scavenge	67.2	0.21	87.60	320	85.01	0.634	69.95%
Samsung Galaxy J5	Local	157.6	0.0743		2121.14		0.599	
	Local+wifi	162.8	0.07668	-3.30	2123.2	-0.09	0.619	
	Scavenge	22.2	0.06967	85.92	318.66	84.98	0.457	23.71%

Table 7. Competitive offloading results of edge detection task.

Device	Test type	Average Battery consumption in mAh	Discharge rate in mAh/sec	mAh Saving %	Average runtime in seconds	runtime improvement in %	Total Energy by Local Execution (Wh)	Total Energy by scavenging [Mobile side + Surrogate side] (Wh)	Total Energy Saving %
Huawei Mediapad	Local	255	0.2114		1206		4.7	3.3327	29.10
	Local+wifi	258.4	0.214	-1.33	1207.52	-0.12			
	Scavenge	103	0.1652	59.61	613.7	49.12			
Huawei Honor 6	Local	257.4	0.1961		1312.65				
	Local+wifi	249.4	0.1787	3.1	1395.4	-6.30			
	Scavenge	64	0.113	75.14	619.45	52.81			
Samsung Galaxy Tab A	Local	542	0.254		2134				
	Local+wifi	543	0.2561	-0.18	2120	0.66			
	Scavenge	93.6	0.21	82.73	461.51	78.37			
Samsung Galaxy J5	Local	157.6	0.0743		2121.14				
	Local+wifi	162.8	0.07668	-3.30	2123.2	-0.09			
	Scavenge	31.6	0.06967	79.95	582.5	72.54			

Results of task 2, scenario 2: Edge Detection of images in competitive offloading

In this competitive offloading scenario, results and evaluation of scavenging for edge detection (task 2) when all the four devices are requesting for the surrogate at the same time is done. The corresponding results are shown in Table 7. These results show that, offloading from multiple clients simultaneously provides a significant advantage for battery saving and performance improvement of mobile devices. The amount of battery charge consumed for task via offloading is only 17% - 40 % of the battery consumption of doing the task locally and the runtime are only 21% to 51% of the runtime of doing the task locally. Not only is offloading advantageous in saving battery life but also it is efficient in terms of discharging rate as it is shown in mAh/sec result column. In addition, an aggregate 29% energy savings are obtained in this case.

An interesting point to observe is that, all of the devices have almost similar performance by offloading as it is shown in almost equal runtime by scavenging. The only exception is Samsung tab A which has a better performance improvement by scavenging. Even though the devices have different performance in executing a task by themselves as it is observed from corresponding runtime result, their performance is enhanced to the same level by offloading. The percentage gains in this scenario are lower than the case for task 2 when the devices are accessing the surrogate alone (scenario 1). This is attributed to the network delay and scheduler queue in sending and receiving processed images to many clients from surrogate.

Results comparison with previous researches

Opportunistic offloading tests using Scavenger were carried out in [13] on 7 inch Samsung Galaxy Tab and Nokia N900 smart phone. The characteristics of these devices are shown in the Table 1 with all other devices. This shows that the devices used in this new experiment are very powerful compared to the devices used in [13]. Even the less powerful device in our experiment (Galaxy j5) is more than twice as powerful compared to the galaxy tab used in [13]. We can also infer that the trend of battery capacity enhancement has been very slow. For instance, the 8 inch Huawei Media pad used in our experiment has 4800mAh capacity and the 7-inch tablet used in previous tests [13] had 4000mAh battery capacity.

A similar task of encryption and decryption a file under local, local execution with wifi (worst case scenario) and remote execution cases are tested on the devices. The surrogate machine used as Intel Core 2 Duo 2.26GHz processor with 2.9 Gb of random access memory. For this task, 89% of battery saving and 83% of runtime savings were recorded for the galaxy tab. For the case of N900, 95% and 96% battery and runtime savings were obtained respectively. In our tests, 87 – 97 % battery savings and 81 – 90% runtime improvements are observed. Another task that were applied for the test in [13] is edge detection of images. However, this task provides a very low advantage in both battery and runtime savings. Only 20% runtime savings and 7% battery saving were measured from the galaxy tab as compared to 80 – 88% battery gains and 75 – 85 % runtime gains measured now. Some of the reasons for better gains in our results are the better surrogate machine used and the improved stability of android platform support for python.

Offloading provides similar and even better benefits with current state of art devices (thus contradicting to our preliminary assumption based on the previous study). The possible reasons that we have identified for this trend are:

- The current trends of having mobile devices equipped with powerful processors and other resources is balanced by increasing powerfulness of potential surrogate machines. For example, the surrogate used in our case is more powerful that surrogate used in [13]. It is twice powerful in memory and is equipped with relatively newer generation of processors.

- The current trends of having mobile devices equipped with powerful processors and other resources is balanced by increasing complexity of the tasks to be processed. For example, complex (high Mpix) images to deal with.

- The improved stability of android platform support for python might also be added to the reasons of why better benefits obtained as compared with results of [13] especially for image edge detection tasks

These battery and runtime benefits are also come with saving of total energy consumed per task in our tests. Through dynamic usage of computation resources, energy efficiency can be improved per computational task. Not only is offloading beneficial when a single device scavenges for resources of surrogates as confirmed by [13] and our results, but also it is beneficial when multiple devices scavenge for resources of surrogates at the same time as it is described in the results of competitive offloading tests.

Threats to validity

This study aims to redo our earlier study, compare results and find implications how new mobile devices have affected the benefits of offloading. This setting aims to ensure the internal validity of the study (design and analysis of results). By using the same cases and tools (only changing the devices) we aimed at consistent and comparable studies. The measurements of the second study were supervised by the person who did the first set of measurements. All this was supervised by the professor. The generalizability of the results (external validity) is still rather limited. This study compares only two different generations of devices from total of 3 manufacturers and as such much more (baseline) measurements are needed for fully generalizable results. Nevertheless we assume that the reliability of the results is valid as the measurements have been done by several persons.

6 Conclusion

The various tests and their corresponding results show that both opportunistic offloading of computation and competitive offloading provide a significant benefit for the mobile side by prolonging battery life and improving their performance. By scavenging for computing resources, the performance of mobile devices can be enhanced as it is observed in our results of series of tests. In addition, it is noticed that energy can also be saved by offloading a computation to a resource rich server. Hence energy consumed per computing task can be minimized by offloading both in opportunistic and competitive scenarios. The current powerful state of art mobile devices can be beneficial by cyber foraging since their increasing computational powerfulness is balanced by the increasing computational power of potential surrogate devices and the increasing complexity (resource demands) of potential mobile applications.

These results also show the potential implication for guiding computing in a sustainable way. The traditional cloud infrastructures are characterized by a very resource rich servers located in relatively few number of datacenters, which are far from potential mobile devices. The possibility of saving energy, battery and runtime by competitive offloading (offloading to servers located close to end devices at the edge of a network) gives us a new possibility to go for a more flexible and dynamic cloud infrastructure which resides close to end devices. More over mobile network providers can also provide such edge infrastructure servers for subscribed customers. The sustainability advantage is not only in saving energy per computation but also the possibility of using mobile devices for a longer duration without the need for new ones. This will reduce the manufacturing life cycle emissions from end devices. However, these also require many number of edge servers to be deployed but as compared to enormous number of end devices, their number is still low and a more optimized hardware and software design can be applied on these specialized edge servers. In addition, the close by servers will enhance the user experience by minimizing latencies. Latencies between clients and a closest public cloud range from 20 – 40 ms and 100 – 150 ms over wired and LTE mobile networks respectively. While this can be acceptable for simple applications such as web browsing, it makes impossible or very difficult to create highly interactive applications. For instance, creating interactive feeling in augmented reality applications require that end-to-end latencies (both processing and networking) remain below 20 ms. Therefore, the availability of these close “mobile clouds” that can execute resource intensive and latency sensitive tasks on behalf of mobile devices enable such applications through minimizing latencies. The rise of Internet of Things devices also increase resource demands from mobile devices. Such devices, e.g. wearable computers are even resource constrained than smartphones and tablets. Therefore, enabling opportunistic offloading will augment such devices resources and enable number of potential applications.

However, to enable the above applications, a more energy and compute resource aware offloading technology or even a protocol level implementation is required. Such implementation has to make the task of deciding and offloading much easier to minimize the overhead due to decision and cost benefit analysis. The Scavenger offloading technology on the other hand requires running the daemon software all the time on the surrogates whether a request from clients is coming or not. Such approach which involves idle daemon running will eventually outweigh or counter balance the energy saving benefits obtained by offloading.

7 Acknowledgments

The research is fully supported by European Union PERCCOM Erasmus Mundus program [16]. The authors would like to acknowledge all the partner institutions, sponsors and researchers of the program.

8 References

[1] M. Satyanarayanan, “Pervasive computing: Vision and challenges,” IEEE Pers. Commun., vol. 8, no. 4, pp. 10–17, 2001.

[2] J. M. Harris, “Sustainability and Sustainable Development,”, International Society for Ecological Economics, 2003.

[3] W. Zhao, Y. Peng, F. Xie, and Z. Dai, “Modeling and simulation of cloud computing: A review,” Proc. - 2012 IEEE Asia Pacific Cloud Comput. Congr. APCloudCC 2012, pp. 20–24, 2012

[4] B. Abbasov, “Cloud computing: State of the art reseach issues,” 8th IEEE Int. Conf. Appl. Inf. Commun. Technol. AICT 2014 - Conf. Proc., 2015.

[5] M. Sharma, H. Bansal, and A. K. Sharma, “Cloud Computing: Different Approach & Security Challenge,” Int. J. Soft Comput. Eng., vol. 2, no. 1, pp. 421–424, 2012.

[6] A. S. Al-Ahmad, S. A. Aljunid, and A. S. A. Sani, “Mobile cloud computing testing review,” Proc. - 2013 Int. Conf. Adv. Comput. Sci. Appl. Technol. ACSAT 2013, pp. 176–180, 2014.

[7] S. Kitanov and T. Janevski, “State of the Art: Mobile Cloud Computing,” 2014 Sixth Int. Conf. Comput. Intell. Commun. Syst. Networks, pp. 153–158, 2014.

[8] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” Futur. Gener. Comput. Syst., vol. 29, no. 1, pp. 84–106, 2013.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The Case for VM-Base Cloudlets in Mobile Computing,” Pervasive Comput., vol. 8, no. 4, pp. 14–23, 2009.

[10] S. Hakak, S.A. Latif, G. Amin, “A Review on Mobile Cloud Computing and Issues in it,” Int. J. Computer Applications, vol. 75, no. 11, 2013.

[11] M. Patel, Y. Hu, P. Hédé, “Mobile-Edge Computing,”, white paper, ETSI, no. 1, 2014, available from: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

[12] G. A. Lewis, P. Lago, and G. Procaccianti, “Architecture strategies for cyber-foraging: Preliminary results from a systematic literature review,” in Proceedings of the 8th European Conference on Software Architecture (ECSA 2014), 2014, pp. 154–169

[13] J. Parkkila and J. Porras, “Improving battery life and performance of mobile devices with cyber foraging,” in IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 2011.

[14] J. Porras, O. Riva, and M. Kristensen, “Dynamic resource management and cyber foraging,” Middleware for Network Eccentric and Mobile Applications, vol. 1, p. 349, 2009.

[15] M. D. Kristensen, “Scavenger: Transparent development of efficient cyber foraging applications,” in Proc. Percom’10, Mannheim, Germany, 2010, pp. 217–226.

[16] Porras J., Seffah A., Andersson K., Rondeau E., Klimova A.: PERCCOM: A Master Program in Pervasive Computing and COMmunications for sustainable development, Conference on Software Engineering Education and Training (CSEE&T), 2016.

APPENDIX 1: Baseline test results

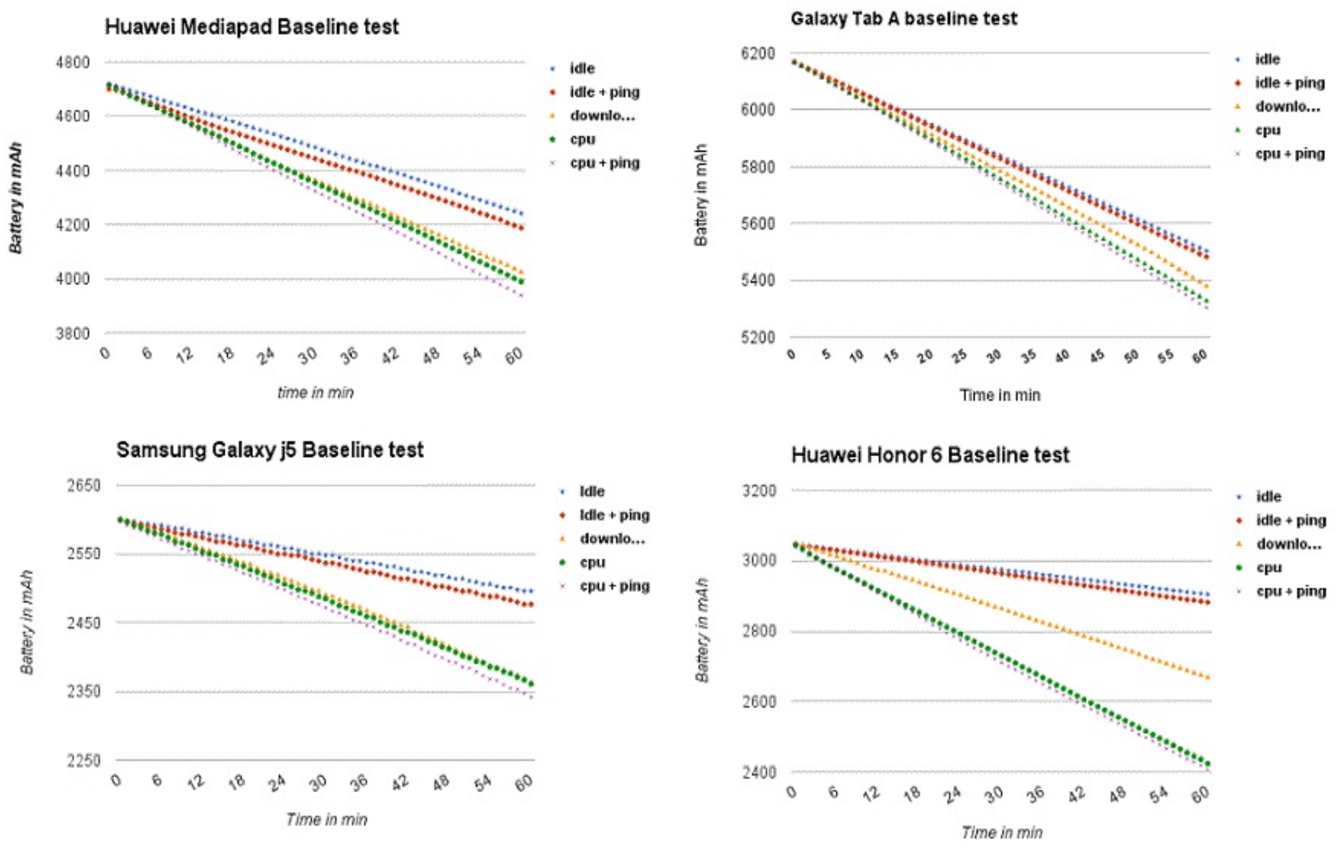

Fig. 1. Baseline test results for the new study

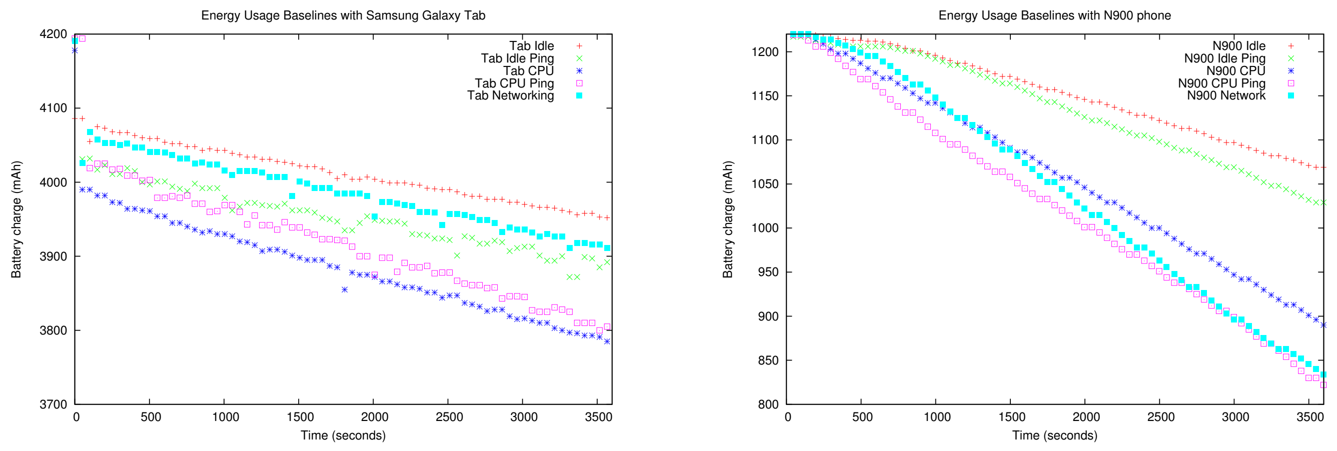

Fig. 2. Baseline test results of the previous study [13]