Ko et al. Journal of Cloud Computing: Advances, Systems and Applications (2021) 10:44
https://doi.org/10.1186/s13677-021-00260-8

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

# Computation offloading technique for energy efficiency of smart devices

Jaejun Ko, Young-June Choi and Rajib Paul*

**Abstract**

The substantial number of wearable devices in the healthcare industry and the continuous growth of the market procreates the demand for computational offloading. Despite major development of wearable devices and offloading techniques, there are several concerns such as latency, battery power, and computation capability that requires significant development. In this paper, we focus on the fact that most smart wearable devices have Bluetooth pairing with smartphones, and Bluetooth communication is significantly energy-efficient compare to 3G/LTE or Wi-Fi. We propose a computation offloading technique that offloads from the smartphone to the cloud server considering the decision model of both wearable devices and smartphones. Mobile cloud computing can elevate the capacity of smartphones considering the battery state and efficient communications with the cloud. In our model, we increase the energy efficiency of smart devices. To accomplish this, a Dhrystone Millions of Instructions per Second (DMIPS)-based workload measurement model along with a computation offloading decision model were created. According to the performance evaluation, offloading from wearable devices to smartphones and offloading once to cloud server can reduce energy consumption significantly.

**Keywords:** Wearable device, Smartphone, Computation offloading, Energy efficiency

## Introduction

Recently, wearable devices has become an integral part of human life and refining several regular activity such as healthy eating, active lifestyle, sufficient exercise, sleep tracking, emergency alert, and many more. Global Information and Communication Technology (ICT) companies are launching different advanced wearable devices; specifically, the healthcare wearable devices are growing every year. According to a report issued by Tractica, a market intelligence agency, in 2016, the market for health care wearable devices is expected to grow rapidly from 2015 to 2021, reaching about $18 billion by 2021 [1]. In addition, CISCO estimates that the number of world-class wearable devices will increase year by year and will reach 924 million by 2021 [2]. Despite a huge number of work on mobile cloud computing, so far very few studies focus specifically

Smart devices such as wearable devices and smartphones have limitations in battery capacity and processor performance that can be mounted with limitations on device size in order to improve portability due to the characteristics of mobile devices. Therefore, there are limitations in performing complex calculations using various sensors such as heart rate, camera, or using network communication service with other devices. To alleviate this problem, the concept of computation offloading has been proposed that transfers complex computation job to cloud server with powerful computing resources. The offloading technique brings several benefits; however, it also hinders the energy consumption performance of limited power devices. In a wireless network considering a bad channel condition, devices can have significant energy consumption to transfer the task to a cloud server. The computation offloading can reduce the energy consumption of wearable devices by using only network communication

*Correspondence: rajib@ajou.ac.kr
Department of Software and Computer Eng, Ajou University, Suwon, South Korea

which consumes less energy than computation processing locally.

The existing computation offloading techniques use a communication module such as Wi-Fi, 3G, or LTE to offload a complex computation task to the cloud server. Most wearable devices have Bluetooth pairing with smartphones, and the energy consumption for Bluetooth is lower than Wi-Fi and 3G/LTE energy [3]; therefore, it is efficient to offload the complicated computation to the smartphone via Bluetooth. Unlike cloud servers with unlimited computing resources, smartphones have limited battery capacity like wearable devices. Therefore, smartphones operating on wearable devices cause high energy consumption and leads to performance degradation. In this paper, we propose a two-step operation offloading technique to select the offloading tasks. This technique considers the energy efficiency of the smartphones during offloading, determines whether the request received from the wearable device should be accommodated according to the current battery state of the smartphone itself or once offloaded to the cloud server. Ultimately, this can improve the overall energy efficiency of smart devices.

We created an energy cost model in a combination of Bluetooth and Wi-Fi communication cost that determines whether to perform computation offloading for smart devices; the workload model is base on the DMIPS proposed in [4]. Also, to evaluate the actual energy efficiency of the proposed method, we implemented the benchmark application with the Hanoi Tower in LG G Watch Urbane W150, Google Nexus 5, and PC, respectively. The changing energy consumption was measured by using Monsoon Power Monitor. We compared the energy efficiency of each smart device and the total energy efficiency when the proposed technique is applied and analyzed the results.

This paper is organized as follows. "Related work" section introduces the existing studies on computation offloading. "Proposed offloading technique" and "Offloading decision model" sections introduce the workload and computation offloading decision models used in the proposed schemes, respectively. Our experimental setup is discussed in "Experiment setup" section. "Results" section, we describe and analyze the results of experiments to prove the energy efficiency of each smart device in the proposed method. Finally, "Conclusion" section presents conclusion and future work.

## Related work

Computational offloading has grabbed a lot of attention among researchers after the novel work of Cuervo [5]. Smart wireless devices [6, 7] bring critical issues like battery life, computation time, and offloading price. These hinders the popularity of wearable devices. In this section, we discuss existing literature studies that relates to our work.

In the past few years, we can find a plethora of work towards offloading problems from mobile devices to the remote server or cloud. Similar to other domains, in cloud computing, Deep Learning (DL) has been widely applied [8–13]. At the same time, a tremendous amount of efforts have been made towards reducing the overload of DL tasks to make it feasible on smartphones [14]. In contrast, wearable devices have much less computation power, and resources compare to smartphones, which makes it very difficult to execute DL tasks. The focus of our work is to improve the energy efficiency of wearable smart devices; therefore, we have mainly focused on papers with similar significance.

An energy-optimal task transmission scheme is proposed in [15] without considering the option of local transmission. For some low-delay tolerance applications, this may worsen the performance. An optimal offloading decision policy is derived in [16] by comparing the energy consumption of local computing. However, the local execution is done only for applications with strict deadlines. Since cloud server has high computational power, we find a similar approach in [17] that refers to the migration of computations to the cloud server. However, the cloud servers are spatially far from devices, which causes high transmission delay and performance deterioration.

In [18], authors have proposed an offloading scheme with a workflow consisting of an assessment phase, scanning phase, selection phase, and offloading phase. Their technique initially discovers a device in the neighborhood and then paired the device with Bluetooth to offload complex tasks with high energy consumption. In addition, they created a communication and computation energy model for offloading decisions, implemented three benchmark applications of OpenGL Cube, Garbage Collection, and SunSpider, and compared the energy consumption when the proposed technique was applied.

Another offloading technique was proposed in [19] that reduces the energy consumption of mobile devices. In the case of processing complex tasks with high energy consumption, it was divided into two processes. Firstly, offloading was performed by Bluetooth communication to a nearby PC, and secondly, offloading was performed by Wi-Fi communication. Energy consumption was compared using the Million Floating Point Operations(MFLOP).

In [20], authors have proposed Jade, a task offloading technique including Profiler, Optimizer, and Communication Manager as components. Jade monitors the state of mobile devices and applications and automatically determines whether it offloads a complex code that consumed high energy to the cloud server. Also, by making a Jade API, developers can easily implement task offloading. Face

detection and FindRoute benchmark applications were implemented, and their energy consumption, CPU load, and Execution Time were measured and compared.

Pomac was proposed in [21] which is a computation offloading technique that offloads Method Invocation Instruction (MII) of applications by intercepting in the Dalvik VM layer. mInterceptor, sMonitor, Classifier, and sInterface were implemented in the Application VM and the offloading decision was made by performing algorithms such as Linear Regression, SVM, NaiveBayes using some features like energy consumption, response time, and network status. Using Android benchmark applications such as DriodSlator, ZXing, and Picaso, they measured the response time and energy consumption in OnDevice, OnServer, Optimal, and Pomac situations.

Author in [22] focused on an energy minimization problem for local computation and task offloading in a single user system. Later this work is extended to a multi-user system that combines both financial costs for instance: network price and energy consumption [23]. Average latency performance was analyzed in [24] including communications delay and computation delay for large-scale spatially random networks. In [25], the authors proposed a hierarchical cloudlet architecture for distributing the workload to reduce the average response time of a task request. They further focused on edge computing-based IoT networks [26]. Conventional offloading techniques are engineered to reduce energy consumption; however, little attention has been paid to legitimate communication module selection.

DeepWear, a deep learning framework was proposed in [27] to improve the performance and reduce energy consumption. It is an intelligent and adaptable offloading strategy to upload from wearable to a paired handheld. However, the author claimed with the limited processing capacity of smartphones DeepWear performs poorly and in addition, no consideration of energy consumption during the training procedure. A context-aware task offloading (CATO) technique was proposed in [28], in which offloading tasks can be properly executed on smartphones or further uploaded based on the context. However, in this framework, wearable devices will always upload to the smartphone if CATO is enabled, which leads to resource underutilization for wearable devices.

In addition to the above-mentioned research, a system is used to offload in [29], that acts as a bridge for computation offloading between a cloud server and mobile device. For details on cloud computing, studies are available on the energy efficiency of mobile devices in [30], and study on classification and the basic structure of offloading technique in [31, 32]. According to the reviewed cloud offloading mechanisms, a side-by-side comparison is presented in Table 1.

## Proposed offloading technique

Unlike cloud servers, which have unlimited computing resources, smartphones have limited resources such as battery capacity. Therefore, when the offloading operation requested by the wearable device is always received and processed by the smartphone, it encounters a usability issue. In this paper, a heuristic algorithm is proposed for a smartphone, whether to accept a computation task uploaded by the wearable device and later offload it to the cloud server. Ultimately, we proposed a novel computation offloading technique that can increase the energy efficiency of smart devices.

The existing offloading schemes have two types of workflow. Firstly, offload to a cloud server when processing a complex computation with high energy consumption, and later the cloud server returns the result. Secondly, the smartphone performs as a primary server, but it is located just between the wearable device. The smartphone only acts as an intermediary medium to transmit the requested operation from the wearable device to the cloud server.

On the other hand, the proposed method includes a process for determining whether to accept a request for the operation of a wearable device in consideration of smartphone energy efficiency and whether to offload the mobile device once to the cloud server.

In Fig. 1, when a specific task is given to a wearable device, Task Analyzer first analyzes the task through the workload model and obtains the task size. After that, we compare the energy consumption with the Bluetooth communication energy cost according to the amount of work and make an offloading decision for the wearable device. According to this proposition, the wearable device will offload the computation task to the smartphone and waits for the result to return.

The smartphone judges whether the request of the wearable device should be accepted or not by comparing the result of the remaining battery energy unit and the energy consumed when processing the local execution. Although the smartphone accepts to execute the task, the smartphone will later offload the same task if it is more energy efficient to once offload it to the cloud server using Wi-Fi communication. If the smartphone can not accept the request of the wearable device, it checks whether offloading is possible to the cloud server. If not, the smartphone sends a message to reject the request. In such a case, a wearable device can use traditional ways to offload. The result processed by each smart device or the cloud server is delivered back to the target wearable device through the Output Manager.

## Offloading decision model

Factors that determine the offloading of smart devices include the local energy consumption to process compu-

**Table 1** Literature study on offloading mechanisms

| Reference | Objective | Environment | Evaluation tools | Wearable devices |
|---|---|---|---|---|
| Shu [15] | Energy | Cloud + Mobile devices | Implementation | No |
| Zhang [16] | Energy within delay period | Cloud + Mobile devices | Simulation | No |
| Songtao [17] | Energy + Delay | Cloud + Mobile devices | Simulation | No |
| Biswajit [18] | Energy | P2P | Implementation | No |
| Kelie [19] | Energy + Delay | Cloud + Mobile devices | Implementation | No |
| Hao [20] | Energy + Delay | Mobile application | Implementation | No |
| Hassan [21] | Energy + Delay | Mobile + Cloud | Implementation | No |
| Jeongho [22] | Energy | Mobile application | Simulation | No |
| Yongjin [23] | Cost + Delay | Cloud + Mobile devices | Simulation | No |
| Seung [24] | Delay | Mobile edge computing | Simulation | No |
| Qiang [25] | Delay | SDN + cloud | Simulation | No |
| Qiang [26] | Delay | Mobile edge computing | Simulation | No |
| Mengwei [27] | Energy + Delay | Wearable + Mobile + Cloud | Implementation | Yes |
| yang [28] | Energy + Delay | Wearable + Mobile + Cloud | Implementation | Yes |

tations and the communication energy costs that result from offloading to the cloud server. At first, we describe the workload format based on the proposed model using DMIPS in our previous studies [4]. Second, we propose computation offloading decision models that consider the communication energy cost that occurs when offloading from each side of each smart device.

### DMIPS based workload model

DMIPS is the value measured by the Dhrystone benchmark application used to measure the performance of the CPU, which means the number of instructions the CPU processes per second [33, 34].

$$W = \frac{DMIPS_m}{U_{dmips}} \times U_{task} \times t_{task} \tag{1}$$

In Eq. (1), $U_{dmips}$ is the CPU utilization when the Dhrystone benchmark application is executed, and $DMIPS_m$ is the median considering the outlier in the VAX MIPS rating value measured using the Dhrystone benchmark application. The workload $W$ can be obtained by multiplying DMIPS per CPU utilization by $DMIPS_m$ and $U_{dmips}$

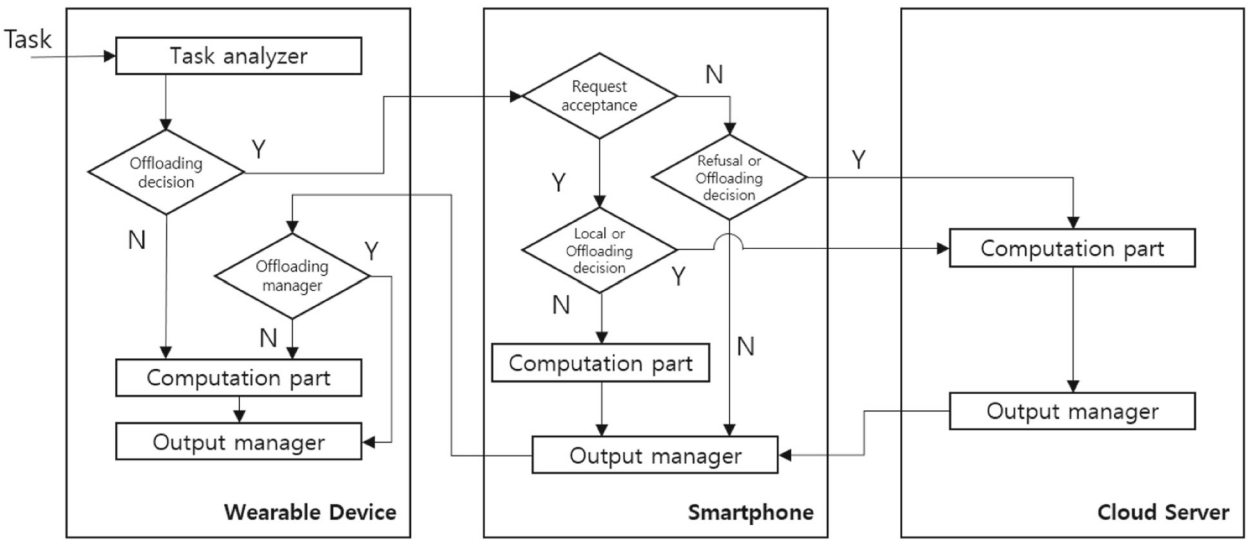



**Fig. 1** Workflow chart of the proposed technique

and by multiplying the CPU utilization $U_{task}$ and time $t_{task}$ at the task execution.

#### Wearable device's decision model

The offloading decision of a wearable device depends on the energy consumption when the computing operation is locally processed in the wearable device and the energy consumption when the work is offloaded to the smartphone with the Bluetooth connection. In Eq. (2), we calculate the amount of energy consumed if the wearable device processes the computation task locally.

$$E_{wear} = E_{W\_wear} \times W \tag{2}$$

In Eq. (2), $E_{W\_wear}$ denotes energy consumed per wear amount of the wearable device, and $W$ denotes the work amount obtained through the workload model. When the wearable device offloads the calculation operation to the smartphone, the energy consumption of the wearable device ($E_{offloading\_wear}$) is the energy consumed for the Bluetooth communication energy cost ($E_{bluetooth}$) and the waiting time until the result is received from the smartphone ($P_{wait\_wear} \times t_{phone}$). Equation (3) is the energy cost model of the wearable device when the wearable device offloads the calculation operation to the smartphone.

$$E_{offloading\_wear} = P_{wait\_wear} \times t_{phone} + E_{bluetooth} \tag{3}$$

$$E_{bluetooth} = \frac{P_{bluetooth}(power)}{T_{bluetooth}(throughput)} \times N(datasize) \tag{4}$$

$$E_{wifi} = \frac{P_{wifi}}{T_{wifi}} \times N \tag{5}$$

In Eq. (3), $E_{bluetooth}$ represents the energy cost of the wearable devices for Bluetooth communication. In the related research [35], energy cost model such as Eqs. (4) and (5) were utilized by considering data size, network throughput, and power consumption in Bluetooth and Wi-Fi communication. In this paper, we have developed an energy cost model that considers Bluetooth communication energy cost. The offloading decision can be determined by comparing energy consumption ($E_{wear}$) of wearable device's locally performed computation task and energy ($E_{offloading\_wear}$) consumption when offloading to the smartphone. We can write as follows:

$$\begin{cases} offloading & \text{if } E_{wear} > E_{offloading\_wear} \\ non-offloading & \text{if } E_{wear} \leq E_{offloading\_wear} \end{cases} \tag{6}$$

#### Smartphone's decision model

When a request for offloading is received from a wearable device, the smartphone first determines whether to accept the request or not. We calculate the energy ($E_{phone}$) consumed when we locally process the operation requested by the wearable device and compare it with the remaining energy of the battery ($E_{battery}$). Equation (7) includes the energy cost for Bluetooth communication to return the result of the calculated operation, i.e., energy consumption when the smartphone locally handles the computation task requested by the wearable device.

$$E_{phone} = E_{W\_phone} \times W + E_{bluetooth} \tag{7}$$

$$E_{battery} = B_{current} \times V \times 3600s \tag{8}$$

Equation (8) is used to determine whether the smartphone can offload, by converting the residue battery of the smartphone into an energy unit. In addition, the smartphone judges whether or not to accept the request of the wearable device. It measures and compares the energy consumption when offloading to the cloud server. Equation (9) calculates the consumed ($E_{offloading\_phone}$) energy when the smartphone offloads the wearable device's request to the cloud server once again and waits until the smartphone receives the results from the cloud server ($P_{wait\_phone} \times t_{cloud}$). Bluetooth energy ($E_{bluetooth}$) and Wi-Fi energy ($E_{wifi}$) are consumed to return to the wearable device and smartphones, respectively.

$$E_{offloading\_phone} = P_{wait\_cloud} \times t_{cloud} + E_{wifi} + E_{bluetooth} \tag{9}$$

In order to improve the energy efficiency of smartphones, the decision to offload is made through the following two comparisons.

$$\begin{cases} offloading & \text{if } E_{offloading\_phone} < E_{phone} \leq E_{battery} \\ non-offloading & \text{if } E_{phone} \leq E_{battery} \leq E_{offloading\_phone} \end{cases} \tag{10}$$

$$\begin{cases} offloading & \text{if } E_{offloading\_phone} < E_{battery} < E_{phone} \\ refusal & \text{if } E_{battery} < E_{phone} \leq E_{offloading\_phone} \end{cases} \tag{11}$$

Upon receiving a request from a wearable device, a smartphone can calculate if the energy consumed ($E_{phone}$) is equal to or less than the remaining energy of the battery ($E_{battery}$). If acceptable, the smartphone will upload to the cloud server. In addition, if the smartphone can not locally process, but the energy cost ($E_{offloading\_phone}$) of offloading to the cloud server is less than the battery residual energy, the smartphone offloads once to the cloud server. However, if the energy consumption during off-loading is greater than the remaining energy of the battery, the smartphone rejects the request of the wearable device.

For a better understanding of our offloading strategy we present a sequential diagram in Fig. 2. A strategic interaction among wearable, smartphone, and cloud through Bluetooth and Wi-Fi is depicted here.

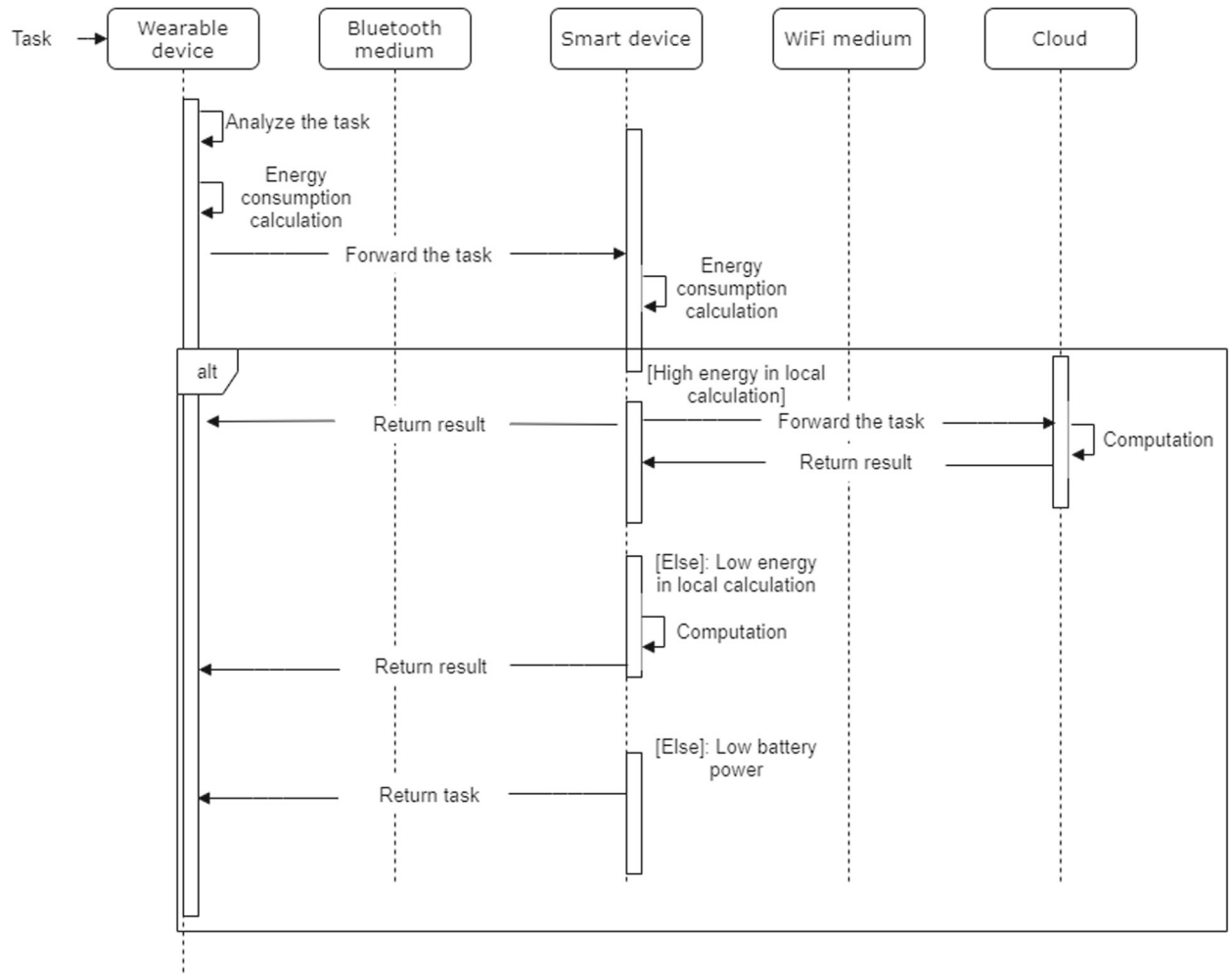



**Fig. 2** Sequential diagram of offloading process

## Experiment setup

### Test devices

The experiment first measured the workload based on the DMIPS and the energy consumption according to the workload change in terms of the wearable device and the smartphone. We used the LG G Watch Urbane Wi-Fi that is a smartwatch based on the Android operating system as a wearable device, and the LG Nexus 5 as a smartphone. We also used a PC instead of a cloud server. Table 2 shows the specifications of the test devices.

### Benchmark applications

The benchmark application used in the experiment is an Android application that handles the Hanoi tower algorithm. The Hanoi tower algorithm exponentially increases the number of recursive function calls as the number of disks increases. Besides, the amount of work done by the CPU increases proportionally, which is the time complexity of the Hanoi Tower algorithm. We chose the Hanoi

**Table 2** Test Device Specification

| | Wearable Device | Smartphone | Cloud Server |
|---|---|---|---|
| Device | LG G Watch Urbane W150 | LG Nexus 5 | PC |
| CPU | Qualcomm Snapdragon 400, 1.2 GHz | Qualcomm Snapdragon 800, 2.26 GHz | Intel Core i7-3770 3.40 GHz |
| Architecture | ARM tortex-A7 | Qualcomm Krait 400 | - |
| Performance | 1.9 DMIPs/MHz | 3.39 DMIPS/MHz | - |
| RAM | 512 MB | 2 GB | 8 GB |
| OS | Andriod Wear | Android Marshmallow 6.0.1 | 645-bit Windows 10 |
| Battery Capacity | 410 mAh | 2300 mAh | - |

Tower as a benchmark application because we can adjust the number of disks without changing other variables and review diverse energy consumption. The Hanoi Tower benchmark application is implemented with two Android applications for smartwatches and smartphones and a Java application for running on PC. Each of them includes Bluetooth and Wi-Fi communication functions and performs arithmetic off-loading with a smartphone or PC.

Along with the Hanoi Tower benchmark application, we also used the Dhrystone benchmark application as a benchmark application to measure the workload of each smart device. We calculate the DMIPS value per CPU utilization by using the VAX MIPS rating value measured by the Dhrystone benchmark application and the CPU utilization at that time, and obtain the workload as the number of disks in the Hanoi Tower increases from each smart device. We use these values to create a workload model.

### Profiling tools

To measure CPU utilization when measuring DMIPS using the Dhrystone benchmark application, we used the Android Debug Bridge (ADB), which can send various commands to Android-based mobile devices.

To measure the CPU utilization and the task execution time of the Hanoi Tower benchmark application, we used the Dalvik Debug Monitor Server (DDMS) of the Android Device Monitor, a profiling tool provided by Android Studios [36]. In the experiment, when the Hanoi Tower benchmark application is executed, the Inclusive Real Time value indicating the working time of the method and the Inclusive Cpu Time% indicating the CPU utilization is measured by using the method profiling of the DDMS.

And also, we used the Monsoon Power Monitor [37], a tool that can analyze the power consumption of all electronic devices using up to 4.55V batteries to measure the varying energy consumption as the number of disks in the Hanoi Tower benchmark application increases with each smart device. Figure 3 shows the connection between the Monsoon Power Monitor and smart devices.

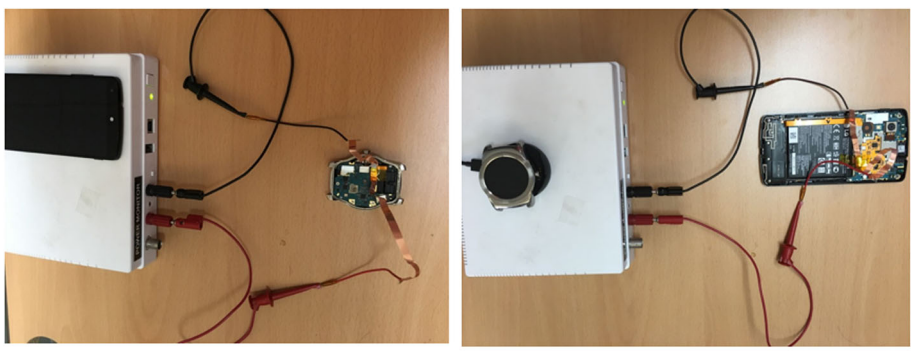
### Method

Experiments were conducted in two parts. One is to measure the CPU utilization of the Dhrystone benchmark application and DMIPS to obtain the workload, the other is to measure the energy consumption of each smart device.

To determine the workload of a smart device, we run the Dhrystone benchmark application 100 times to measure the actual performance of the smartphone. In this case, we use the ADB command (adb shell top -s cpu -m 5 -d 0.1) to profile the top five CPU utilization rates of the processes running on the smartphone, so that the CPU utilization of the Dhrystone benchmark application. As the number of disks in the Hanoi tower increases, the workload time required to obtain the changing workload and the CPU utilization during the work are measured using DDMS, and nonlinear regression analysis is performed based on the collected data to create a workload model for the experiment.

The energy consumption of each smart device is measured based on three scenarios. In each, the number of disks in Hanoi Tower increases from 13 to 23, and we have used Monsoon Power Monitor to measure energy consumption. In the first scenario, a wearable device (smartwatch) processes computation tasks locally, and the energy consumption of the smartwatch ($E_{wear}$) is measured. In this scenario, the result is expressed in terms of $S1.Wear$ for convenience.

The second scenario is when a smartwatch offloads a task to a smartphone and is processed locally by the smartphone. In this scenario, the energy consumption of smartwatches ($E_{offloading\_wear}$), which includes the Bluetooth communication energy cost, and the energy consumption



(a) LG G Watch Urbane Wi-Fi (b) LG Nexus 5

**Fig. 3** Connection between the Monsoon Power Monitor and Smart Devices

**Table 3** Experiment Scenario

| Scenario | Content | Measurement | |
|---|---|---|---|
| 1 | Local execution of wearable device. | S1.Wear | $E_{wear}$ |
| 2 | Offloading a computation task to smartphone, and the smartphone executes locally. | S2.Wear | $E_{offloading\_wear}$ |
| | | S2.Phone | $E_{phone}$ |
| 3 | Offloading a computation task that wearable device offloads to the cloud server once again. | S3.Wear | $E_{offloading\_wear}$ |
| | | S3.Phone | $E_{offloading\_phone}$ |

of the smartphone ($E_{phone}$) to execute the computation tasks requested by smartwatch are measured. For convenience, the energy consumption of the smartwatch and the smartphone is expressed as $S2.Wear$ and $S2.Phone$, respectively.

The third scenario involves the smartwatch first offloading a computation task and offloading the smartphone to the cloud server once again. In this scenario, the enaergy consumption of the phone ($E_{offloading\_phone}$) including smart watch's energy consumption ($E_{offloading\_wear}$), Bluetooth, and Wi-Fi communication energy costs is measured. As for the other scenarios, the energy consumption of smartwatches and smartphones is expressed as $S3.Wear$ and $S3.Phone$, respectively. Table 3 shows a description of each experimental scenario.

## Results

The proposed technique is to offload complex computation tasks with high energy consumption from a wearable device to a smartphone or a cloud server to increase the energy efficiency of the smart device. In the proposed method, the smartphone accepts the request of the wearable device or offloads it to the cloud server considering its current battery status and energy efficiency. In this section, we first show the measured workload based on DMIPS. Later analyze the energy consumption results of each smart device measured by Monsoon Power Monitor in three experimental scenarios, and compare the total energy efficiency.

### Workload measurement

In order to calculate the workload per Hanoi tower disk in the smartphone, ADB measured the CPU utilization rate and found the median value as 48.596%. The median value of DMIPS measured by the Dhrystone benchmark application was 3266. Along with these two values, we obtained the workload ($W$) of the smartphone (LG Nexus 5) by substituting the CPU utilization measured by the DDMS function of Android Device Monitor and the task execution time into the workload model formula (1). Figure 4 shows the workload of LG Nexus 5 according to the number of disks in Hanoi Tower.

#### *Time complexity*

Figure 4 shows that the workload increases exponentially with the time complexity ($O(2^n)$) of the Hanoi Tower algorithm. According to the Hanoi Tower algorithm, we can forward only one task at a time from wearable to the smartphones or smartphone to cloud server. Based on this, a non-linear regression analysis was performed using the measured results to create a workload model

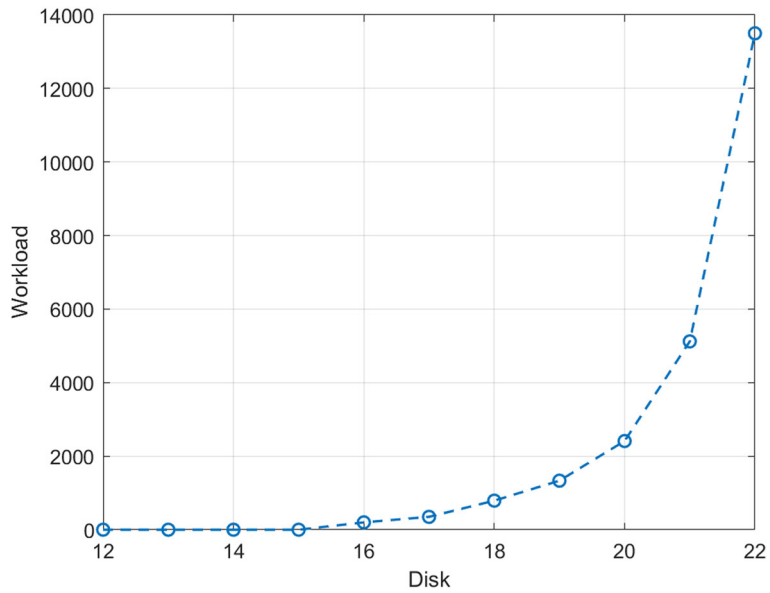



**Fig. 4** The workload of the LG Nexus 5 according to the number of disks

to be applied to the experiment. In our previous study [4], DMIPS, CPU utilization, and task execution time differ according to the CPU performance of each smart device, and as a result, the workload models are similar to each other. Therefore, we can analyze that the workload of the LG G Watch Urbane Wi-Fi, which is the wearable device used in the experiment, is not different from the LG Nexus 5 workload. Table 4 shows the results of the non-linear regression analysis of LG Nexus 5.

#### Energy consumption measurement

The energy consumption of each smart device is measured based on three experimental scenarios. In Scenario 1, the energy consumption ($S1.Wear$) of the smartwatch is measured when the complex computation with high energy consumption was processed locally by the smartwatch. Scenario 2 is the case where smartwatch offloads computation tasks to a smartphone and the smartphone process it locally. The energy consumption of the smartwatch ($S2.Wear$) and the energy consumption of the smartphone ($S2.Phone$) is measured. In Scenario 3, smart watch's energy consumption ($S3.Wear$) and smartphone energy consumption ($S3.Phone$) is measured when the smartwatch offloaded the computation task offloaded from the smartphone to the cloud server once again. Figure 5 shows the energy consumption of the smartwatch ($S1.Wear$, $S2.Wear$, $S3.Wear$) in each experimental scenario and we can easily understand that offloading to the cloud server can stabilize the energy consumption.

As shown in Fig. 5, in Scenario 1, the energy consumption ($S1.Wear$) of the smartwatch increases exponentially as the number of disks in the Hanoi Tower benchmark application increases. In Scenario 2, the increase in the energy consumption of the smartwatch ($S2.Wear$) is similar to when the task is executed locally in Scenario 1 ($S1.Wear$). However, since the performance of a smartphone that performs computation tasks of a smartwatch is relatively good compared to that of a smartwatch, we can observe a 77% reduction compare to the maximum disk in Scenario 1. In addition, the energy consumption of Bluetooth communication to offload computation tasks, energy consumed waiting for the return of operation result, consumes more energy than when smartwatch executes computation operation locally in a section with fewer disks. In Scenario 3, despite the increase in the number of Hanoi tower disks due to the superior performance of the cloud server, the amount of energy consumed by the smartwatch is almost constant. The energy consumed by the smartwatch includes the energy consumed for a short period until the result of the offloading operation is returned and the Bluetooth communication energy cost consumed in offloading.

**Table 4** The non-linear regression analysis

| Formula | | Workload$\sim a * \exp(b * Disk) + c$ | | | |
|---|---|---|---|---|---|
| **Device** | **Var** | **Estimate** | **Std. Error** | **T Value** | $Pr(>\|t\|)$ |
| LG Nexus 5 | a | $9.991e^{-02}$ | $4.154e^{-02}$ | 2.405 | 0.0428 |
| | b | $6.16e^{-01}$ | $1.813e^{-02}$ | 33.986 | $6.14e^{-10}$ |
| | c | $-1.093e^{+03}$ | $8.490e^{+02}$ | −1.288 | 0.2339 |

Figure 6 compares the energy consumption of the smartwatch ($S1.Wear$, $S2.Wear$, $S3.Wear$) as the number of Hanoi towers increases in the three experimental scenarios. From 16 Hanoi tower disks, the smartwatch offloads computation tasks to smartphones, which means less energy is consumed. Also, since the number of disks in Hanoi Tower is 18, it should be offloaded to the cloud server, but in the $16-18$ sectors, it needs to be processed in the smartphone to consume less energy.

Figure 7 shows the energy consumed as the workload ($W$) of the smartwatch obtained from the regression model in the three test scenarios increases. If the workload of the smartwatch is larger than the intersection point ($\alpha$) between $S1.Wear$ and $S2.Wear$, offloading to the smartphone without processing the computation operation is less energy-consuming. On contrary, if it is larger than the intersection point ($\beta$) of $S1.Wear$ and $S3.Wear$, it is better to offload the task requested from the smartwatch once again to the cloud server in terms of the energy efficiency of the smartwatch. Besides, if the intersection point ($\gamma$) of $S2.Wear$ and $S3.Wear$ is larger than that of the smartwatch, the energy consumption of smartwatches can be reduced by processing computation in a cloud server rather than a smartphone. According to this, no optional strategy can be adopted for offloading, rather the network should strategically decide to offload.

As shown in Fig. 8, in Scenario 2, the energy consumption of the smartphone ($S2.Phone$) increased exponentially, similar to the smartwatch energy consumption measurement in ($S2.Wear$). However, on average smartphones consume more energy than smartwatches because their screens are much larger than a smartwatch and it comprises energy costs for returning results via Bluetooth communication.

The energy consumption of the smartphone measured in Scenario 3 ($S3.Phone$) includes the cost of Wi-Fi communication energy consumed to offload the computation job once again to the cloud server, the energy consumed during the waiting for the result to return. And Bluetooth communication energy consumed when returning results to a smartwatch.

Figure 9 compares the energy consumption of smartphones ($S2.Phone$, $S3.Phone$) due to the increase in the number of Hanoi tower disks in Scenarios 2 and 3. From the number of disks in Hanoi Tower is 17, it is better to offload a task requested from a smartwatch

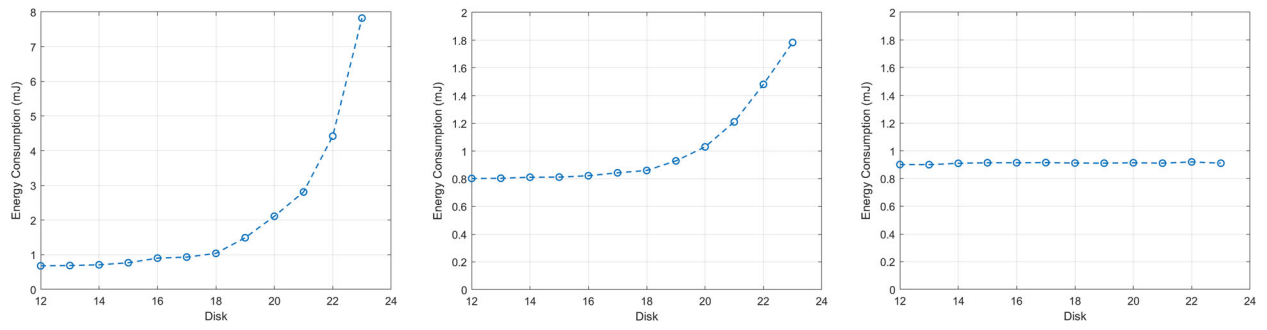



**Fig. 5** Energy Consumption Measurement Results of Smart Watch

to the cloud server once on the smartphone's energy efficiency side.

Figure 10 is a graph showing the energy consumed by the smartphone as the workload ($W$) obtained from the regression modeling method increases in scenario 2 and 3. If energy is consumed by the smartphone ($S2.Phone$) to locally execute the computation operation is smaller than the current energy level of the battery, i.e., $E_{battery} = 0.1$, the smartphone accepts the offloading request of the smartwatch. At this time, it can be seen that if the workload of the smartphone is larger than the intersection ($\alpha$) between $S2.Phone$ and $S3.Phone$, offloading to the cloud server is better in terms of the energy efficiency of the smartphone. Although the energy ($S2.Phone$) consumed by the smartphone is larger than $E_{battery}$, when the energy consumed for offloading the operation once to the cloud server is less than $E_{battery}$, the energy consumption can be reduced.

### Energy efficiency comparison of smart devices

In the previous section, our experimental results demonstrated that the energy consumption of smart devices in each scenario increases with the number of Hanoi tower disks (or workload). When the number of disks is 17, the

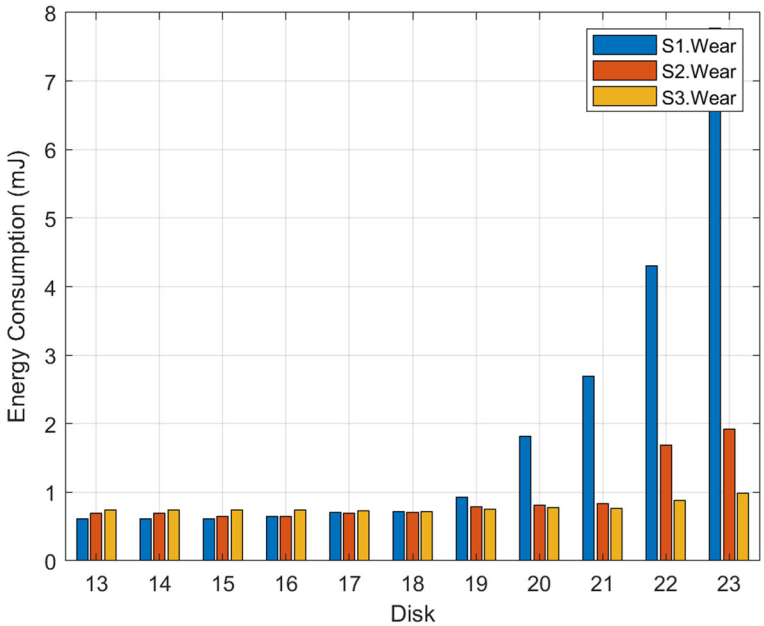



**Fig. 6** Comparing the energy consumption of Smart Watch according to the number of disks

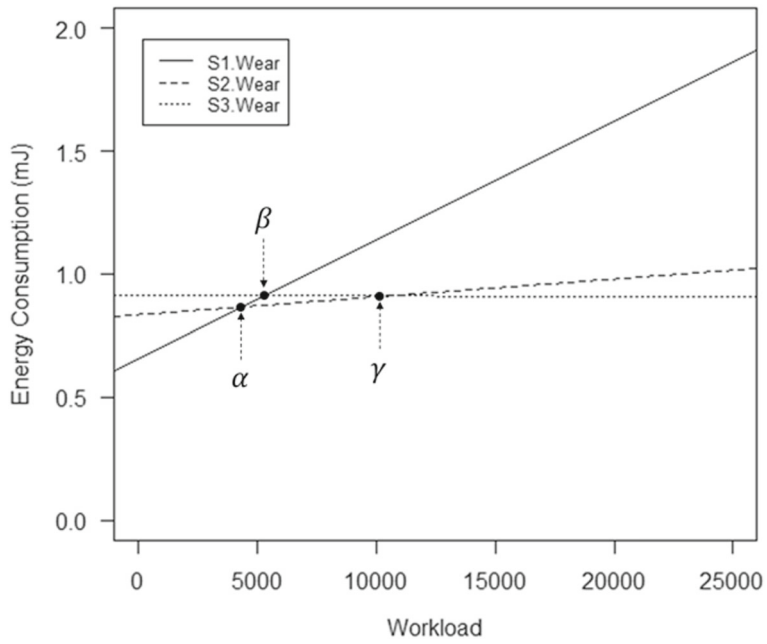



**Fig. 7** Energy consumption of Smart Watch according to workload

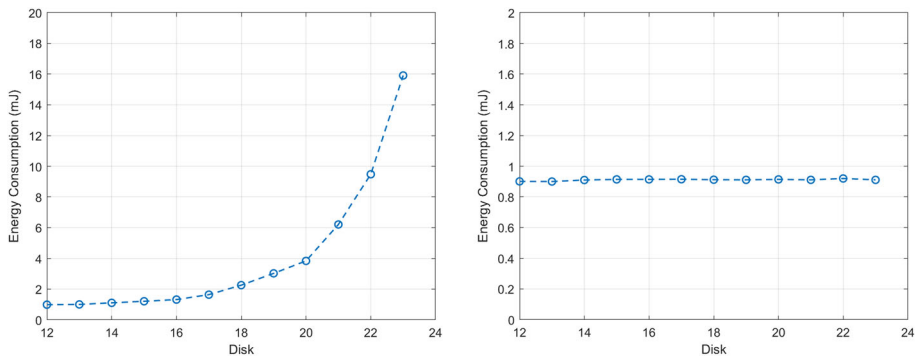
smartwatch consumes less energy when the offloading operation is not offloaded once to the cloud server and processed by the smartphone. At this time, offloading one more time to the cloud server from the smartphone side is more energy efficient.

However, as you can see in Fig. 11, the overall energy consumption of the smartphone is lower than the one offloading to the cloud server. This is due to the energy consumed by the smartwatch for Bluetooth communication to offload the computation task and the energy consumed to complete the computation task in the cloud server.

## Conclusion

In this paper, we proposed a computation offloading technique to solve the problem of low usability of smart devices caused by small battery-powered wearable devices.

The existing computation offloading techniques are offloading from a smartphone to a cloud server via Wi-Fi, 3G/LTE network environment, or offloading to a cloud server using a smartphone as a simple intermediary in a wearable device. In contrast, the proposed technique uses a smartphone as a server that accepts offloading of a wearable device, offloads it once to the cloud server in




(a) S2.Phone (b) S3.Phone

**Fig. 8** Energy Consumption Measurement Results of Smartphone

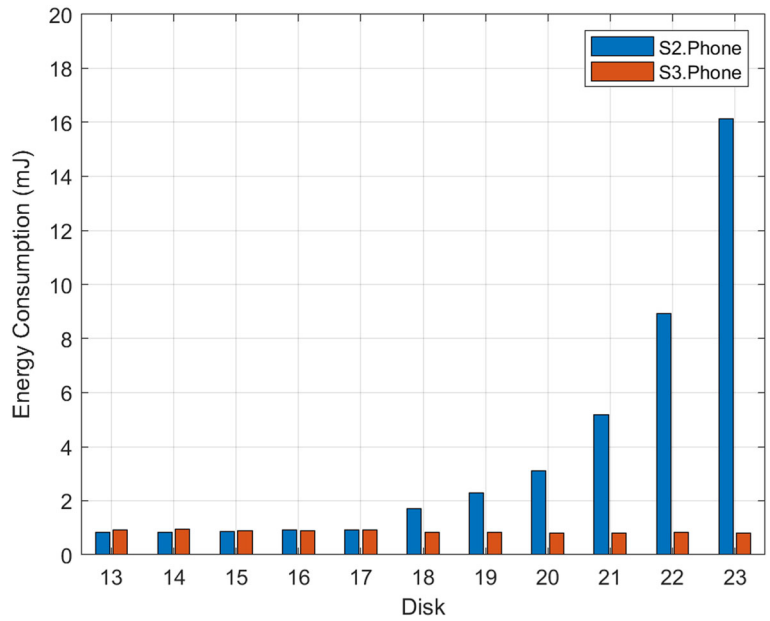



**Fig. 9** Comparison of energy consumption of Smartphone according to number of disks

consideration of the battery status and energy efficiency of the smartphone.

In this paper, we dealt with a workload model based on DMIPS and created the energy cost model including Bluetooth, and Wi-Fi communication energy cost to determine the computation offloading of a wearable device and smartphone. To evaluate the actual efficiency of the proposed method, the Hanoi Tower benchmark application was implemented. The local energy consumption of each smart device and the total energy consumption of the smart device were measured and analyzed based on the three experimental scenarios. As a result, we confirmed that offloading from wearable devices to smartphones from a certain point and offloading once to the

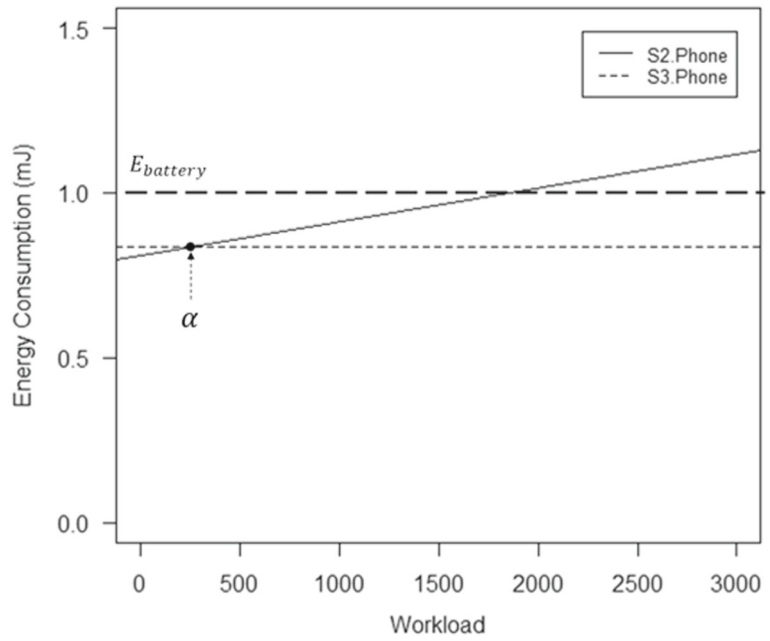



**Fig. 10** Comparison of energy consumption of smartphone according to workload

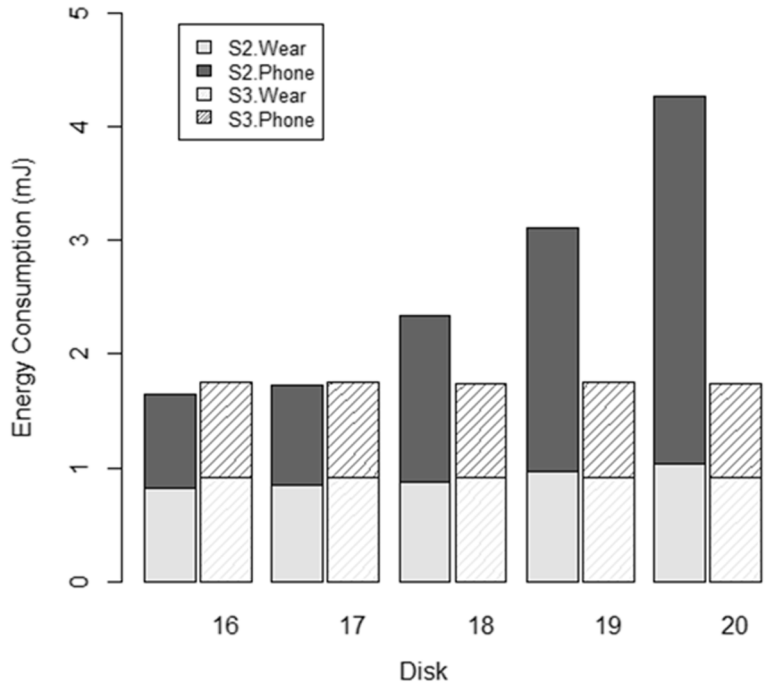



**Fig. 11** Total energy consumption of smart devices

cloud server is low energy consumption on the side of the wearable device. On the smartphone side, it is possible to accommodate requests from wearable devices, but offloading once to the cloud server is energy efficient.

When we compared the local energy consumption and the total energy consumption of each smart device from the experimental results, it is evident that offloading a task requested from the wearable device to the cloud server without locally processing is not good in terms of energy efficiency; however, the total energy efficiency is better.

**Acknowledgements**
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1008530).

**Authors' contributions**
Conceptualization, J.K, R.P, and Y.-J.C.; formal analysis, J.K.; funding acquisition, Y.-J.C.; methodology, R.P, R.B. and Y.-J.C.; project administration, Y.-J.C.; resources, Y.-J.C.; software, R.B.; supervision, Y.-J.C.; validation, R.P.; writing–original draft, RP and J.K.. All authors have read and agreed to the published version of the manuscript.

**Authors' information**
**Jaejun Ko**: Jaejun Ko received B.S and M.S degrees in Computer Enginerring from Ajou University, South Korea in 2015 and 2017 respectively. He is currently working as a Researcher at Tmaxsoft Inc. His research interests include the power management for smart devices. E-mail: goj609@gmail.com.
**Young-June Choi**:Young-June Choi is a professor at Ajou University, Korea. He received his B.S., M.S., and Ph.D. degrees from the Department of Electrical Engineering and Computer Science, Seoul National University, Korea, in 2000, 2002, and 2006, respectively. From Sept. 2006 through July 2007, he was a postdoctoral researcher at the University of Michigan, Ann Arbor, MI, USA. From 2007 through 2009, he was with NEC Laboratories America, Princeton, NJ, USA, as research staff member. He joined Ajou University from Sept. 2009 as a faculty member and founded Intelligence Platform, Network, and Security Lab. He is an adjunct professor at Seoul National University from Sept. 2015 and a honorary professor at Xiangtan University, China from Dec. 2013 through 2015. Prof. Choi is serving as General Vice-chair of WCNC 2020, served TPC chair of ICOIN 2019, and chaired for many international conferences. He has published more than 150 international papers and holds 50 patents. He received the best paper award from the Journal of Communication and Networks in 2015, the Haedong Young Scientist Award in 2015 and many academic awards. He is an IEEE senior member, an executive director of Korean Institute of Communications and Information Sciences (KICS) and a director of Korean Institute of Information Scientists and Engineers (KIISE).
**Rajib Paul** : Rajib Paul is an Assistant Professor at Ajou University, Korea. He received B.TECH Degree on Electronics and Communication Engineering from West Bengal University of Technology, India in 2009. He acquired his Maters and Ph.D. degree from the Department of Information and Computer Engineering, Ajou University, in 2012, and 2017, respectively. His research focus includes cognitive radio networks, wireless communications, vehicular networks, machine learning, cloud computing, and device-to-device communications.

**Funding**
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1008530).

**Availability of data and materials**
Not applicable.

## Declarations

**Competing interests**
The authors declare that they have no competing interests.

Received: 20 November 2020 Accepted: 29 July 2021
Published online: 21 August 2021

**References**

1. Omdia (2016) Healthcare Robotics. Informatech, Howick Place, London. https://omdia.tech.informa.com/OM011974/Healthcare-Robotics

2. CISCO (2019) Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022. https://s3.amazonaws.com/media.mediapost.com/uploads/CiscoForecast.pdf
3. Li X, Zhang X, Chen K, Feng S (2014) Measurement and analysis of energy consumption on android smartphones. In: 2014 4th IEEE International Conference on Information Science and Technology. IEEE. pp 242–245
4. Lee J, Ko J, Choi Y-J (2017) Task offloading technique using DMIPS in wearable devices. In: 2017 International Conference on Information Networking (ICOIN). IEEE. pp 414–416
5. Cuervo E, Balasubramanian A, Cho D.-k., Wolman A, Saroiu S, Chandra R, Bahl P (2010) Maui: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. pp 49–62
6. Seneviratne S, Hu Y, Nguyen T, Lan G, Khalifa S, Thilakarathna K, Hassan M, Seneviratne A (2017) A survey of wearable devices and challenges. IEEE Commun Surv Tutor 19(4):2573–2620
7. Sun H, Zhang Z, Hu RQ, Qian Y (2018) Wearable communications in 5g: challenges and enabling technologies. IEEE Veh Technol Mag 13(3):100–109
8. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach. J Netw Comput Appl 178:102974
9. Farahbakhsh F, Shahidinejad A, Ghobaei-Arani M (2021) Multiuser context-aware computation offloading in mobile edge computing based on Bayesian learning automata. Trans Emerg Telecommun Technol 32(1):e4127
10. Shahidinejad A, Ghobaei-Arani M (2020) Joint computation offloading and resource provisioning for e dge-cloud computing environment: A machine learning-based approach. Softw Pract Experience 50(12):2212–2230
11. Shakarami A, Ghobaei-Arani M, Shahidinejad A (2020) A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. Computer Networks 182:107496
12. Jazayeri F, Shahidinejad A, Ghobaei-Arani M (2020) Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach. J Ambient Intell Humanized Comput 12:1–20
13. Saeik F, Avgeris M, Spatharakis D, Santi N, Dechouniotis D, Violos J, Leivadeas A, Athanasopoulos N, Mitton N, Papavassiliou S (2021) Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. Comput Netw 195:108177
14. Liu S, Lin Y, Zhou Z, Nan K, Liu H, Du J (2018) On-demand deep model compression for mobile devices: A usage-driven model selection framework. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. pp 389–400
15. Shu P, Liu F, Jin H, Chen M, Wen F, Qu Y, Li B (2013) eTime: Energy-efficient transmission between cloud and mobile devices. In: 2013 Proceedings IEEE INFOCOM. IEEE. pp 195–199
16. Zhang W, Wen Y, Guan K, Kilper D, Luo H, Wu DO (2013) Energy-optimal mobile cloud computing under stochastic wireless channel. IEEE Trans Wirel Commun 12(9):4569–4581
17. Guo S, Xiao B, Yang Y, Yang Y (2016) Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. IEEE. pp 1–9
18. Patra B, Roy S, Chowdhury C (2015) A framework for energy efficient and flexible offloading scheme for handheld devices. In: 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS). IEEE. pp 1–6
19. Zhan K, Lung C-H, Srivastava P (2014) A comparative evaluation of computation offloading for mobile applications. In: 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom). IEEE. pp 518–525
20. Qian H, Andresen D (2015) Reducing mobile device energy consumption with computation offloading. In: 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). IEEE. pp 1–8
21. Hassan MA, Bhattarai K, Wei Q, Chen S (2014) {POMAC}: Properly offloading mobile applications to clouds. In: 6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)
22. Kwak J, Kim Y, Lee J, Chong S (2015) Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J Sel Areas Commun 33(12):2510–2523
23. Kim Y, Kwak J, Chong S (2017) Dual-side optimization for cost-delay tradeoff in mobile edge computing. IEEE Trans Veh Technol 67(2):1765–1781
24. Ko S-W, Han K, Huang K (2018) Wireless networks for mobile edge computing: Spatial modeling and latency analysis. IEEE Trans Wirel Commun 17(8):5225–5240
25. Fan Q, Ansari N (2018) Workload allocation in hierarchical cloudlet networks. IEEE Commun Lett 22(4):820–823
26. Fan Q, Ansari N (2018) Application aware workload allocation for edge computing-based IoT. IEEE Internet Things J 5(3):2146–2153
27. Xu M, Qian F, Zhu M, Huang F, Pushp S, Liu X (2019) Deepwear: Adaptive local offloading for on-wearable deep learning. IEEE Trans Mob Comput 19(2):314–330
28. Yang Y, Geng Y, Qiu L, Hu W, Cao G (2017) Context-aware task offloading for wearable devices. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN). IEEE. pp 1–9
29. Khairy A, Ammar HH, Bahgat R (2013) Smartphone energizer: Extending smartphone's battery life with smart offloading. In: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE. pp 329–336
30. Shi C, Habak K, Pandurangan P, Ammar M, Naik M, Zegura E (2014) Cosmos: computation offloading as a service for mobile devices. In: Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing. pp 287–296
31. Altamimi M, Abdrabou A, Naik K, Nayak A (2015) Energy cost models of smartphones for task offloading to the cloud. IEEE Trans Emerg Top Comput 3(3):384–398
32. Kumar K, Liu J, Lu Y-H, Bhargava B (2013) A survey of computation offloading for mobile systems. Mob Netw Appl 18(1):129–140
33. Arora M, Kalra M, Singh S (2013) Acof: Autonomous computation offloading framework for android using cloud. In: 2013 2nd International Conference on Information Management in the Knowledge Economy. IEEE. pp 143–149
34. York R (2002) Benchmarking in context: Dhrystone. ARM, March
35. Friedman R, Kogan A, Krivolapov Y (2012) On power and throughput tradeoffs of wifi and bluetooth in smartphones. IEEE Trans Mob Comput 12(7):1363–1376
36. Android Using DDMS. https://android-doc.github.io/tools/debugging/ddms.html
37. Monsoon Solution Inc. (2015) Mobile Device Power Monitor Manual Ver 1.15, Bellevue. https://docplayer.net/7313840-Mobile-device-power-monitor-manual-ver-1-15.html


## Publisher's Note