Paper

Mobile Cloud for Parallel and Distributed Green Computing

Leszek Siwik, Dawid Kala, Mateusz Godzik, Wojciech Turek, Aleksander Byrski, and Marek Kisiel-Dorohinicki

AGH University of Science and Technology, Krakow, Poland

https://doi.org/10.26636/jtit.2017.111817

Abstract—Mobile Computing and Mobile Cloud Computing are the areas where intensive research is observed. The "mobility" landscape (devices, technologies, apps, etc.) evolves so fast that definitions and taxonomies do not catch up with so dynamic changes and there is still an ambiguity in definitions and common understanding of basic ideas and models. This research focuses on Mobile Cloud understood as parallel and distributed system consisting of a collection of interconnected (and virtualized) mobile devices dynamically provisioned and presented as one unified computing resource. This paper focuses on the mobile green computing cloud applied for parallel and distributed computations and consisting of outdated, abandoned or no longer needed smartphones being able to set up a powerful computing cluster. Besides showing the general idea and background, an actual computing cluster is constructed and its scalability and efficiency is checked versus the results obtained from the virtualized set of smartphones. All the experiments are performed using a dedicated software framework constructed in order to leverage the no-longer-needed smartphones, creating a computing cloud.

Keywords—distributed computing, green computing, mobile cloud, mobile computing, parallel computing, pervasive computing.

1. Introduction

The omnipresence of mobile devices, smartphones, tablets and wearable devices calls for proposing new ways of leveraging their computing power especially that smartphones or phablets with hexa- or octa-core CPUs on board are not a rarity on the market. In fact, they are becoming the standard. Unfortunately, their immense computational power, unbelievable for the users of the Desktop PC only a little more than two decades ago, is utilized (wasted in fact) for such primitive tasks as social network integration, multimedia production, reality enhancement, etc.

Apparently, mobile gaming is a more demanding application, (especially with 3D and VR processing) but similarly to a situation where GP GPU devices or gaming consoles have been adapted to solving computing tasks [1], the question arises if the enormous computational power of mobile devices can be applied for scientific applications such as parallel and distributed computing. Obviously, there are some restrictions and limitations (the most trivial is the battery lifetime) and a singular mobile device is not so powerful to be applied for scientific processing, but what if we put hundreds or thousands of mobile devices together to set up an (ad hoc) cluster of computing devices, in order to perform certain computations? That is a very attractive and interesting idea, especially taking into consideration the number of available devices which together can potentially provide a really significant computational power. Additionally, mobile devices are "located in the environment" and equipped with sensors so during computation additional "observations" or measurements can be performed. What is more, mobile devices are always in constant and direct contact with the user, which significantly extends possible applications of such computational unit(s).

This is how the, so-called Ubiquitous Computing [2], Mobile Computing and Mobile Cloud Computing (MCC) fields of research came about and the latter two are also in the limelight of our research.

MCC is a kind of combination of mobile devices, (cloud) computing and (rich) communication. Recently, intensive research in this area has been observed but there is still ambiguity in definitions and common understanding, especially given that the market of mobile devices, technologies and apps evolves so fast that definitions and taxonomies do not catch up so dynamic changes.

In 2010 in [3] Mobile Computing was defined as "information at fingertips anywhere, anytime". We disagree, since meeting only one single condition that the user is constantly interconnected using his device and is able to open a browser, social media app or any other communication app to check out or to gain information about anything just under their "fingertip" is not a mobile computing, it is just Internet access "at fingertips".

So, what is mobile (cloud) computing? Well, the proper starting point for defining Mobile Cloud are Cloud and Cloud Computing notions. According to [4], Cloud "*is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers*".

Simultaneously, according to [5], Cloud Computing "*is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management*".

On the basis of the above definitions in [6] Mobile Cloud Computing is defined as "*a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle*".

The definition describing in the best way our research and perspective is the one of Cloud presented in [4] but in our case, it has to be adjusted taking the mobility into account. So, following [4] we define Mobile Cloud as *(a type of) parallel and distributed system consisting of a collection of interconnected (and virtualized) mobile devices dynamically provisioned and presented as one unified computing resource*.

Obviously, this requires the addition of a section saying that the resources are provisioned "*according to the service-level agreements established through negotiation between the service provider and consumers*" but since in our research we leave these aspects aside and focus on volunteer model, this part can be omitted in the definition.

Since, as mentioned above, intensive research in mobile (cloud) computing area has recently been performed, different models and approaches have been proposed [7]–[9].

Utilization of mobile devices in computing usually assumes that some of the computing tasks (or even most of them) will be off-loaded to a dedicated computing infrastructure connected wirelessly (see [10], [11]). An interesting concept of bi-location in an agent-based mobile cloud is presented in [12]–[13], where the actual configuration of the infrastructure may be perceived and efficiently mapped to the control-layer utilizing the notion of agency. A similar approach to make the computing application more portable is based on embedding the computing tasks into so-called weblets and deploying them in the mobile cloud [14]. An interesting realization of the Map-Reduce programming model on handheld devices is presented in [15]. A dedicated framework for supporting mobile-computing, using REST was also discussed in [16].

As one may see, researchers address and focus on many different aspects of mobile computing i.e. integration of "traditional cloud" with mobile devices [17], [18], code offloading [19], [20], energy wasting and battery lifetime [21], integration of cloud computing model with Internet of Things [22], frameworks for (mobile) distributed processing [23], security and user privacy [24]–[26], sensor utilization [27], heterogeneity [6], etc.

In this case, we focus on a green computing cloud [28] applied for parallel and distributed computations consisting of outdated, abandoned or no longer needed smartphones being able to set up a powerful computing cluster.

One has to keep in mind that the predicted number of smartphones should rise to the immense figure of 6.1 billion worldwide by 2020 [29], and producers will continue to tempt users to constantly replace their devices with newer versions, condemning the older ones, often still in a good shape, to extinction. Thus, scavenging the remains of the contemporary social civilization can lead to the construction of a truly renewable, easily configurable and powerful, yet completely "green" computing hardware.

To visualize the scale of wasted computing power, let us briefly analyze the issue. On the basis of our experiments, the average mobile device, computational power can be estimated as 0.37 GFlops. Assuming that the billion new phones introduced into the market every year replace old phones, we can conclude that at least 10 PFlops of potential computing hardware is wasted every year. This value is comparable with supercomputers from the top of the Top500 list [30].

In [31] a comprehensive study of leveraging the mobile devices in order to set up a green-computing appliance is presented. We follow and extend this idea exploring scalability and efficiency of both: a homogeneous and heterogeneous mobile computing cloud.

In Section 2 the concept of the framework is presented, followed by the development issues encountered during the implementation process. Next, the experimental verification of the appliance is shown, based on solving selected benchmark problems in a distributed environment. Finally the paper is concluded and future work is sketched.

2. Mobile Cloud Platform for Green Computing

Mobile Cloud Platform (MCP) is a platform for performing parallel computations on strongly distributed, heterogeneous yet massively available and increasingly powerful mobile devices used as a self-manageable computational unit. The main goal and task of the platform is making it possible to set up a cloud of mobile devices constituting a heterogeneous volunteer computational environment. An important assumptions is that when the (part) of computational task is realized on the device(s), it is possible to use their sensor if it is required or helpful for completing the task. Obviously, devices working on the same computational task can communicate with each other, distributing some parts of the task, collecting (partial) results, requiring some additional activities (using sensors), etc.

Two main conceptual assumptions of presented MCP are:

- self-manageability – understood here as the possibility of setting up the cloud, defining a new task, running tasks and collecting results without any additional servers,
- cross-platforming – the cloud can be set up of any mobile devices running any operating system (iOS, Android, Windows Phone, etc.).

Bringing the big picture of our mobile cloud idea it is assumed that the cloud that is set up takes the form of the hierarchical tree where every single node has n branches at most (see Fig. 1). The root node – in the center of the graph – is connected with five nodes, that form five different

branches of mobile devices. Some of them have further branches attached, etc.

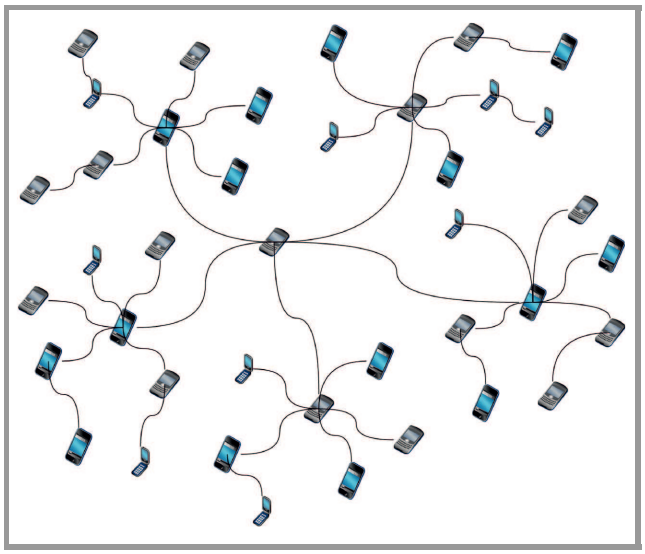

Fig. 1. Hierarchical structure of the mobile cloud.

Keeping the hierarchical structure in mind, the main operations on the cloud, i.e. joining the cloud and starting the computational task, are realized as follows on the conceptual level:

- joining the cloud by the device – the device willing to join the cloud sends a message to any of the devices being already part of the cloud. The message is being delegated to the device being the root of the cloud's tree. The root makes a decision where in the cloud's structure the new device should be joined. The decision is made taking into account the number of devices located in all (sub)trees. When the device has joined the cloud, the root device sends it information about its direct root;
- starting a computational task on the cloud – the device willing to start a new computational task on the cloud sends the message about the new task to all its neighbors. Next the message about the new task is (recursively) propagated among all the devices constituting the cloud. The device which is not able to send the message any further sends back the information if it is going to become the computational node for the new task or not. The message including the list of all the devices going to take part in solving the new task is being sent back to the devices that originated the new task. Next the device originating the new task sends the information to all the devices going to work on the new task that they are part of the virtual tree working on the new computational job.

As one may notice, the aforementioned assumptions require that the direct peer-to-peer communication among devices is available, which requires that either the devices are working within the same LAN or that all devices are publicly available (in the networking sense) so e.g. that they all utilize IPv6 protocols and addressing. There are some restrictions in this context at the moment since most of devices on the market utilize IPv4 protocols but for instance the T-Mobile operator made the IPv6 configuration as the default configuration on the U.S. market for all new devices with Android 4.3 or above from 2013. In December 2013, the list of devices configured for IPv6 included among others [32]: Samsung Galaxy Note 3, Galaxy Light, MetroPCS Samsung Mega, and Google Nexus 5.
Now, the set of IPv6 pre-configured devices looks as follows [32] (hit by December, 12th, 2016): Samsung Galaxy S5, HTC One M8, Samsung Note 3, Samsung Galaxy Light, Google Nexus 5, MetroPCS Samsung Galaxy Mega, Samsung Galaxy S4, Samsung Galaxy Note 2, Google Nexus 4, Samsung Galaxy S3 (latest firmware only), Samsung Galaxy S2 (with Android 4.0 update).
It means that the restrictions for mobile devices direct communication are going to be gradually eliminated and more and more devices available on the market will support the peer-to-peer communication. It is also worth remembering that more and more devices are constantly available in Wi-Fi networks so the direct communication restriction is also (at least partially) eliminated depending on the network configuration.

3. Selected Realization and Implementation Aspects

The easiest way to explain and show the most important realization aspects will be to present how the two most important operations, i.e. joining the cloud by the new device and starting new task, are realized.

3.1. Joining the Cloud by the New Device

When the new device wants to join the mobile cloud it first has to send *ConnectDeviceRequest* message to any device already being part of the cloud (see Fig. 2). *ConnectDeviceRequest* contains among others the IP address of the accessing device.

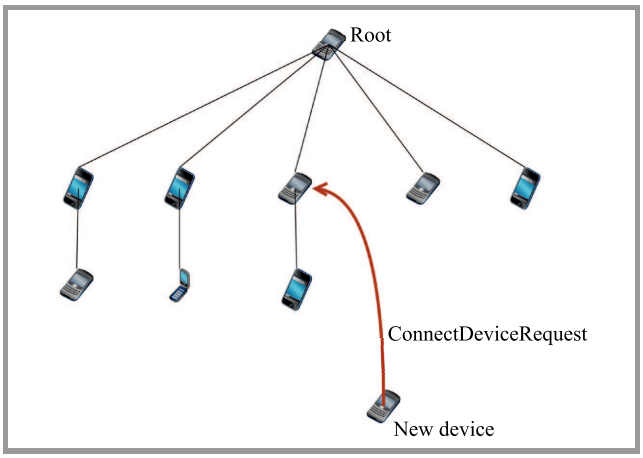

Fig. 2. Connect device request.

Next the device which received *ConnectDeviceRequest* propagates the message up to device being the root of the cloud at the moment.
Next, on the basis of the information about the number of devices accessible through each of its direct children, the root device propagates the *ConnectDeviceRequest* down to the subtree with the lowest number of devices.
The device becoming the direct parent of accessing device sends to the new device the *SaveDeviceConnectionRequest* (see Fig. 3). *SaveDeviceConnectionRequest* message contains among others the IP address of the parental device.

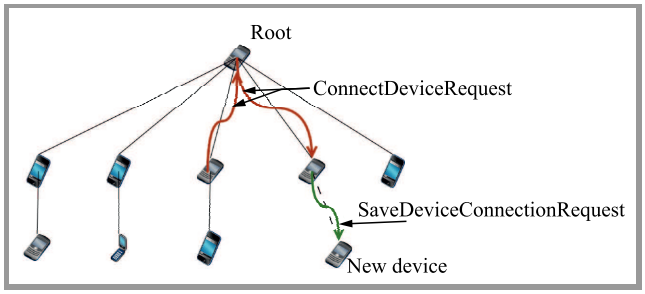

Fig. 3. Save device request.

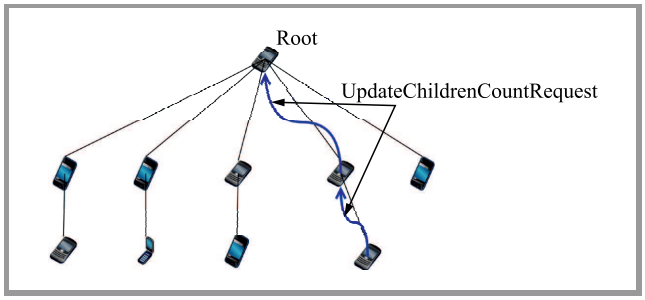

Fig. 4. Update children count request.

Finally, the newly connected device sends up to the parent (which is further propagated up to the root device) the *UpdateChildrenCountRequest* message (see Fig. 4). Every time, the message is sent up to the parental node, it contains the total number of devices in all subtrees on a given level. This way, whenever a new device attempts to access the cloud the cloud knows where it should be connected at the given cloud configuration to achieve a balanced number of devices in all subtrees.

3.2. Starting a New Task on the Cloud

Starting a new task to be computed on the cloud is realized as follows.
Firstly, the device willing to start the new task sends *StartTaskExecutionRequest* message. The message contains among the others: definition of the task to be run (the source code or meta definition), requirements regarding (geo)localization of the devices where the task should be run, information about sensors required to run the task on the single device etc.

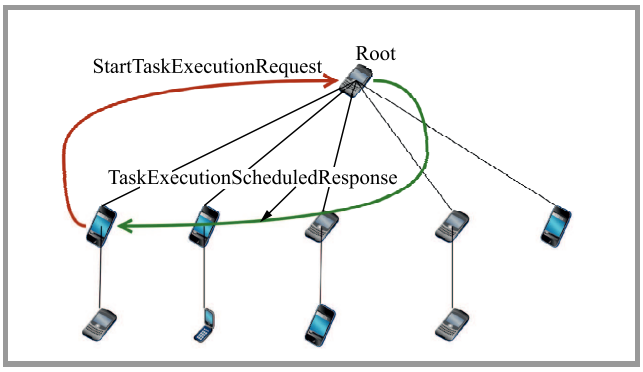

Fig. 5. Start task execution request.

When the device being part of the cloud receives the *StartTaskExecutionRequest* message, the unique UUID task identifier is set up and the *TaskExecutionScheduledResponse* message with the task identifier (see Fig. 5) is sent back.

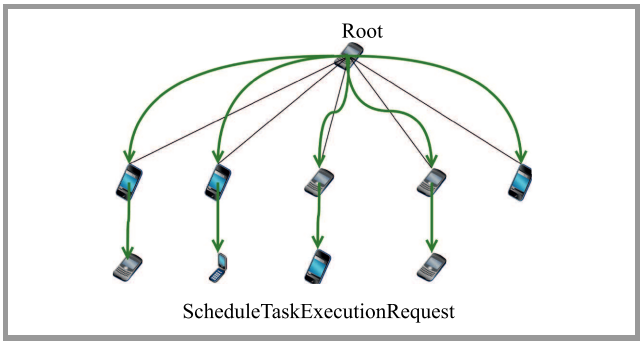

Fig. 6. Schedule task execution request.

Next, the device which has received the *StartTaskExecutionRequest* message propagates the message about the new computational task to all the neighbors as *ScheduleTaskExecutionRequest* message. *ScheduleTaskExecutionRequest* message is propagated through the whole cloud (see Fig. 6).
During the *ScheduleTaskExecutionRequest* message propagation, on the basis of the filters and required features contained in the task definition (required geolocation, sensors etc.) every single device makes a decision whether to participate in the realization of the task or not.
Devices that are not able to propagate the message any further (it is the leaf which did not initialize the propagation process) send the *RemoteTaskReadyRequest* message back to the device they received the *ScheduleTaskExecutionRequest* from.
This part of communication allows for propagation of the information about the devices which have accepted the new task. Along with propagation of the *RemoteTaskReadyRequest* through the cloud – in the message there are gradually aggregated identifiers (IP addresses) of the devices which decided to compute the new task (see Fig. 7).
The last part of starting and configuring the new task on the cloud is propagating the information about the topology of the devices participating in solving the new task.

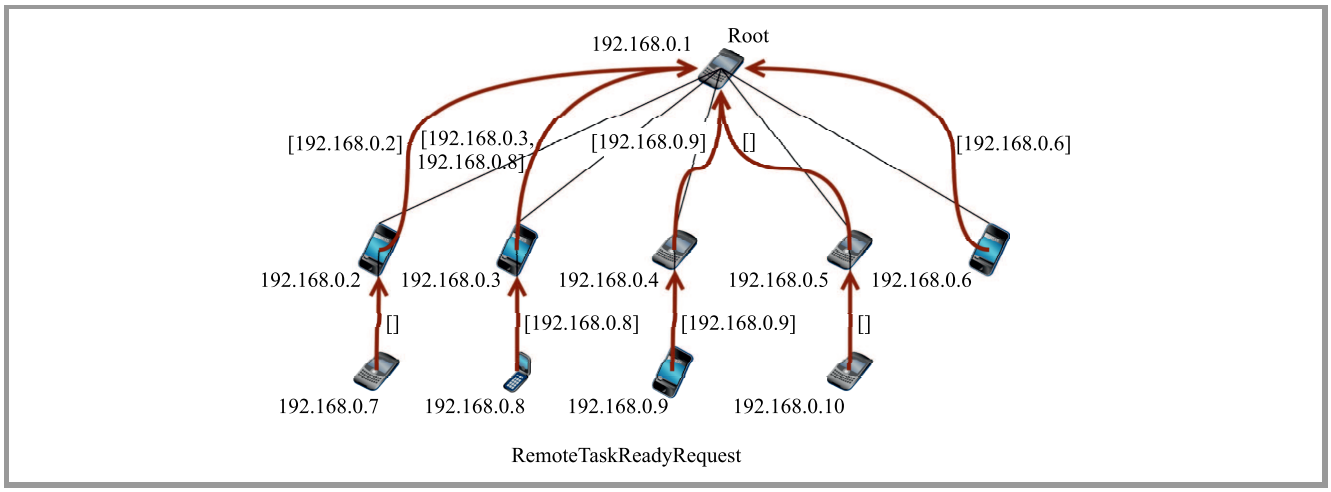

Fig. 7. Task ready request.

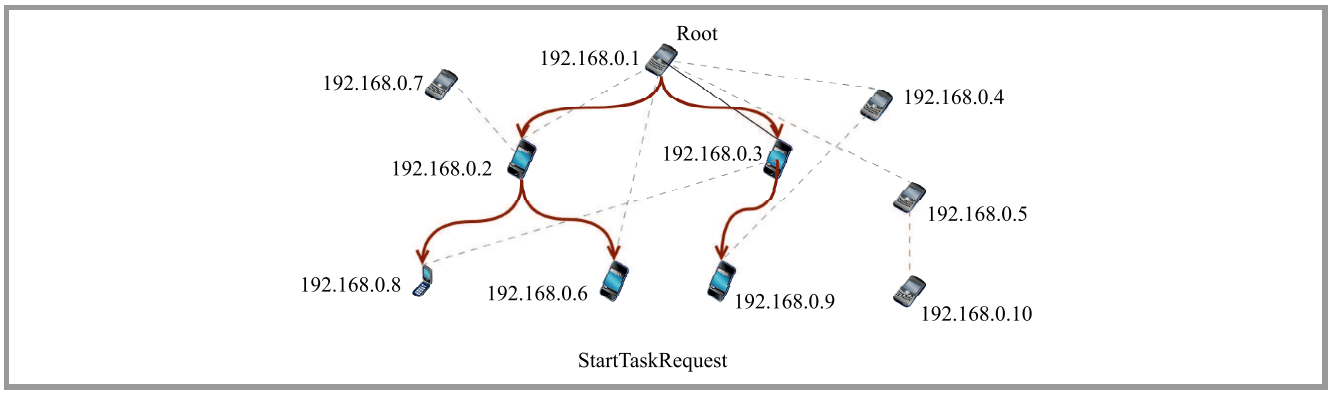

Fig. 8. Start task request.

It consists in creating a temporary, binary tree topology containing all devices which accepted realization of the new task. During this process, within aforementioned temporary topology, the *StartTaskRequest* message is propagated. The *StartTaskRequest* message contains an ordered collection of IP addresses of all the devices taking part in the realization of a new task (see Fig. 8). When the *StartTaskRequest* message is received, the device has all the information about the task definition and the configuration of the task realization environment so it is ready to start and run the task.

4. Experimental Results

The platform has been implemented and tested to verify experimentally its capacity to fulfill functional requirements, ability to perform distributed computations, and to assess its computational power.

The experiments involved heterogeneous devices of different brands, equipped with different hardware and controlled by different operating systems. Therefore, we are not hoping to get fully linear speedup of computations. However, showing that a cloud of cheap devices consuming very little energy can provide significant computational power seems to be a valuable result.

4.1. Functional Verification

The first stage of experiments was the functional verification, i.e. confirming that the platform, all its components, communication patterns and protocols work properly.

This experiment was performed using virtual mobile devices since it is easier to manage a set of virtual devices of the same type than a set of heterogeneous real devices and since using virtual rather than real devices does not negatively interfere with functional verification.

The first step was setting up a sample cloud. It was done using one of arbitrarily selected virtual devices running a simple user interface, thus making the process easier.

Next, we gradually added next devices to the cloud one by one. Devices can be joined to the cloud either by the graphical user interface on one of the devices being available in the same network or by the device that wants to join the cloud.

Next, the experiment of π estimation by Monte Carlo method with $3 \cdot 10^9$ evaluated points was repeated on the cloud consisting of a growing number of the same virtual Android mobile devices. The same experiment but with $3 \cdot 10^{10}$ number of points was also conducted. The times of computations in both cases are presented in Fig. 9.

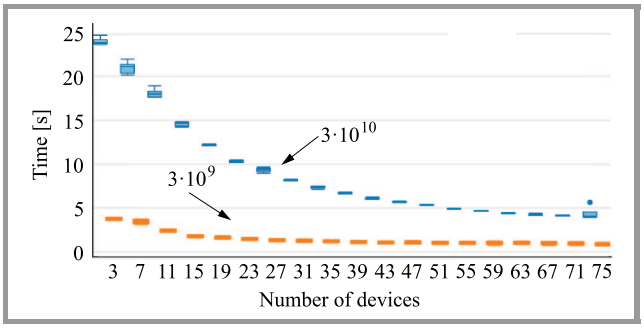

Fig. 9. Time of computations with a growing number of virtual mobile devices.

Since there are no differences among hardware and software specifications of the devices taking a part in computations, the classic picture of shortening computation time along with the growing number of computational units (i.e. virtual android mobile devices) was observed.
Figures 10 and 11 presents the speedup $S = \frac{T_1}{T_N}$ and the efficiency $E = \frac{S}{N}$ for estimating π by the Monte Carlo method with $3 \cdot 10^9$ and $3 \cdot 10^{10}$ number of points. The first conclusion from presented results is that since devices are the same, very typical characteristics of the speedup and the efficiency can be observed. The second conclusion is that the more difficult problem to be solved is, the higher the profit from the parallelization can be observed since when the instance of the problem is to small, communication overhead consumes potential profits from the parallelization.
The task executed during above experiments is relatively simple but at this point the goal was not to solve highly sophisticated or difficult problem but to confirm that all the components of the platforms work properly and that it is possible to set up a mobile cloud and to run computational tasks over there. The general conclusions coming from the experiments presented in this subsection is that a proper functioning of the proposed idea and implementation of particular parts of the platform and communication protocols could be observed and was confirmed. If so, the next step was setting up the cloud consisting of real heterogeneous mobile devices and running some computational tasks in it.

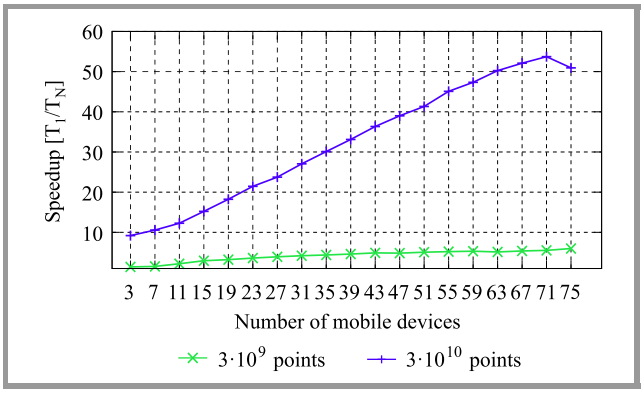

Fig. 10. Speedup as the function of the number of mobile devices for π value estimation by the Monte Carlo method with $3 \cdot 10^9$ and $3 \cdot 10^{10}$ points.

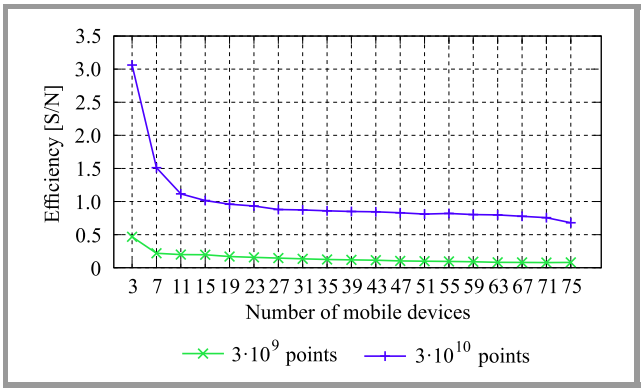

Fig. 11. Efficiency as the function of the number of mobile devices for π value estimation by the Monte Carlo method with $3 \cdot 10^9$ and $3 \cdot 10^{10}$ points.

4.2. Experimental Verification on Real Mobile Devices

In order to perform the tests with real devices, we collected several old and unused devices from our colleagues. We managed to collect 14 devices, however 5 of these failed to execute the client application of the platform or did not start at all. This is the first among the important issues which must be expected when building a system of this type. The main aim, that is putting together of the computing system out of trash, is an appealing idea. However, as it will be made clear, some real drawbacks were spotted. Of course, they were mostly caused by the technical condition of the collected hardware.

The first set of tests aimed at estimating the computational capabilities of particular devices. Each device formed a cloud composed of itself only. The task was again to determine of the π number by random selection of points within a square and a circle (Monte Carlo method). The number of points was $3 \cdot 10^9$. The tests were repeated 5 times. The results are presented in Fig. 12.
Significant differences in performance were expected to be observed (again, because of the fact that these devices were randomly collected). However, this simple experiment shows another major issue related to utilization of this kind of hardware. Six out of nine tested devices give repeatable results, which makes it possible to predict their behavior. The other 3 devices showed a very significant dispersion between different test runs. This was probably caused by the lack of stability of the operation systems loaded with different software installed earlier by the user. For the next experiments, we will thoroughly clean up the hardware and make it homogeneous from the point of view of operating system. This might be easily done using e.g. Cyanogen Mod[1] that removes all the unnecessary "bloat-ware" installed by the developer or the dealer. Moreover, this mod spans over multiple hardware configurations, so installation

[1]CYNG, Company Industries Computer Software Palo Alto, CA, https://cyngn.com

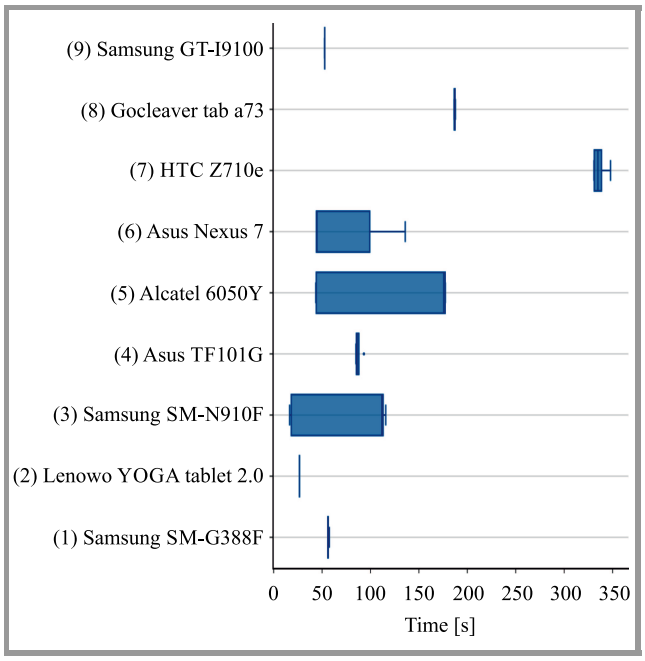

Fig. 12. Evaluation of performance capabilities of the considered devices.

of recent versions of Android systems can be realized even if the version is not supported using the older hardware.

In order to verify that, the computing platform can provide a significant increase in performance, a set of tests with a growing number of devices was performed. The computational task was the same as previously (estimating the value of π with the Monte Carlo method and with the number of points set to $3 \cdot 10^9$). The parallelization was achieved by assigning each device an equal subset of points to verify, so the tasks were of the same size.

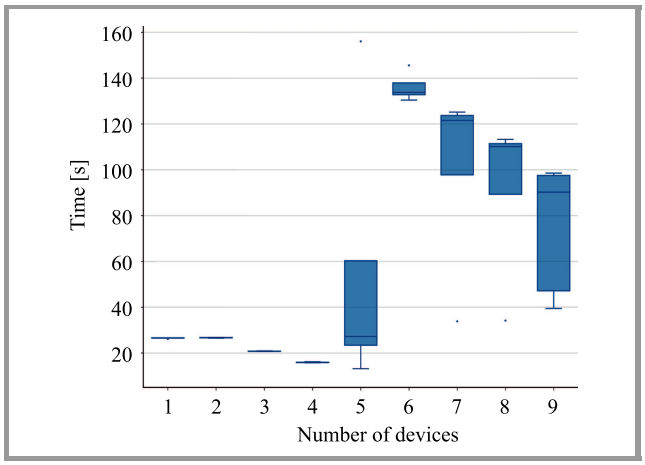

Fig. 13. Time of computations with a growing number of devices, starting from the fastest one.

The first approach was to add new devices, starting from the fastest one down to the slowest according to performance results presented in Fig. 12 – the order was: 2, 9, 1, 6, 3, 4, 5, 8, 7. The results are presented in Fig. 13.

Clearly, adding the third and fourth device increased the performance of the computing cloud. Adding the device no. 3 – Samsung SM-N910F – results in a huge drop in performance.

In the second set of tests, the devices were added in the inverted order. The results are shown in Fig. 14.

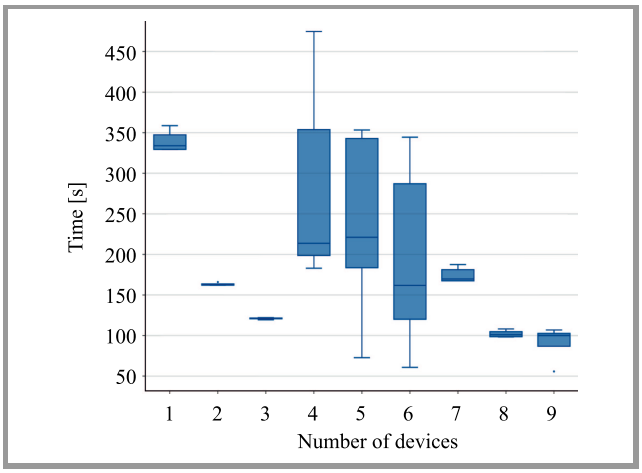

Fig. 14. Time of computations with growing number of devices, starting from the slowest.

Similarly, adding the second and the third device significantly decreases computing time. Along with the growing number of devices, a higher dispersion of results and a significant drop in performance can be observed. This issue can be addressed by extending filters in the task definition with minimal required computational power to be met by each device going to be a part of the computational task. This way modern devices with powerful CPUs would perform computing operations, whereas old devices would provide e.g. measurements and "observations" from available sensors.

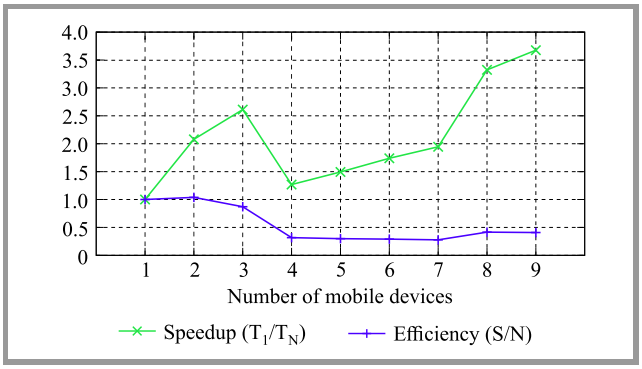

Fig. 15. Speedup (T_1/T_N) and efficiency (S/N) as the function of the number of real mobile devices in the cloud.

Figure 15 shows the speedup $S = \frac{T_1}{T_N}$ and the efficiency $E = \frac{S}{N}$ of the cloud set up from (obsolete) real devices. This time, in contrast to speedup and efficiency for the homogeneous environment presented in Figs. 10 and 11, characteristics are not so regular due to cooperation between

the devices with completely different hardware and software specifications so with totally different computational power and time needed to complete the assigned (part of the) task.

4.3. Estimating Mobile Cloud Computational Power

An important question one may ask regards the computational power of the cloud of cooperating mobile devices set up according to the model presented in this paper.
To estimate the power, the algorithm for π estimation was developed in C++ with MPI. Next, π estimation task was run separately on a single PC station with Intel i7-3370K CPU and on the cloud of the following twelve physical devices: HTC Sensation Z710e, JSR Soul, TCT Alcatel One Touch (x2), Samsung Galaxy Nexus, Samsung GT-I9505, Samsung GT-I9300, Samsung GT-P5100, Samsung GT P5313, LGE Nexus 4, Sony C6603, HTC Desire X. During the experiments task duration on both environments for the same problem sizes wasmeasured and used for further estimations.

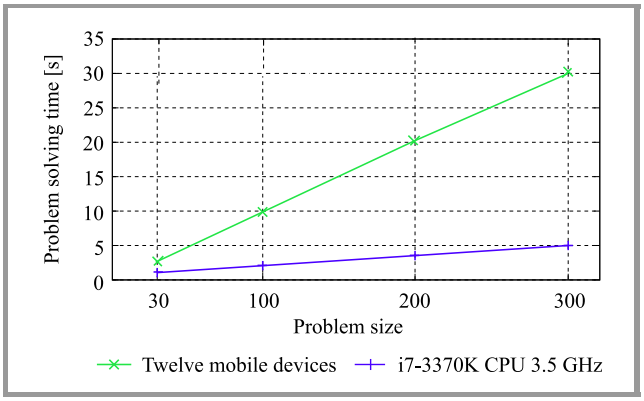

Fig. 16. Time of π estimation with the Monte Carlo method on mobile cloud and PC station with Intel i7-3370K @3.50 GHz CPU.

The results obtained on the mobile cloud and the PC station for 100, 200 i 300 million of points are presented in Fig. 16. Taking the average values obtained on both computational environments, it is possible to calculate coefficient M representing the efficiency ratio between mobile cloud and PC station.

$$M \approx \frac{1.75+3.60+5.35}{3} \div \frac{(10.10+20.25+30.61)\times 12}{3} \approx 0.0146$$

Using IntelBurnTest v2.54[2] the computational power of Intel i7-3370K CPU was set as 25.47 GFlops. So, taking both: the calculated computational power of the CPU used in the experiment and the calculated value of M coefficient, the computational power of a "theoretical" single mobile device working in the presented mobile cloud can be estimated at ~0,37 GFlops.

[2]IntelBurnTest is a tool for measuring CPU efficiency. It is available under: http://www.majorgeeks.com/files/details/intelburntest.html.

For comparison,

- to generate the same computational power as Tianhe-2 supercomputer [33][3] (33.9 PFlops) a mobile cloud consisting of 91.6 million mobile devices would have to be set up,
- to generate the computational power comparable to the power of the Bitcoin network c.a. 6.3 trillion mobile devices would have to be used which would be hardly possible,
- to generate the computational power comparable to the power of the BONIC grid c.a. 158.4 million of mobile devices would have to be used.

5. Conclusions and Future Work

Putting together a number of mobile devices, recovered for instance "from trash", makes it possible to set up easily a computing cloud that may be used for solving many different computational tasks – although the hardware setup and the operational system related issues may appear (and they did, as it was presented in this paper). Keeping in mind the rapidly growing computational power of single mobile device(s), the possible applications of such an appliance are indeed very broad. Devices with octa-core CPUs were not rarity even in 2014 [35] and today they are rather the standard [36].
The concept of collecting and connecting a large number of such devices in order to provide significant computational power seems valuable. The cost of building such a computing cluster is relatively low. Moreover, the cost of computations is also low – the devices are designed to consume as little energy as possible.
The most natural application of the proposed platform is of course volunteer computing leveraging donors' devices, however in this case two fundamental questions may be asked:

- what is the "business model" of the solution i.e. why should people agree to utilizing their (personal) devices for performing some external, "exotic" (computational) tasks, potentially blocking out sensors, consuming the battery power, consuming mobile data bandwidth etc. Well, it is for sure an important question but the same questions are valid in the case of any volunteer computing platform and since the goal of this paper is to present the concept, some realization aspects and preliminary experimental verification, it is out of its scope to discuss the business model;
- what is the type of tasks that could be (or even should be) efficiently run on a mobile cloud platform. Well, the number of mobile devices available on the market

[3]World's fastest supercomputer according to the TOP500 lists from June 2013 until November 2015. It was surpassed in June 2016 by the Sunway TaihuLight [34].

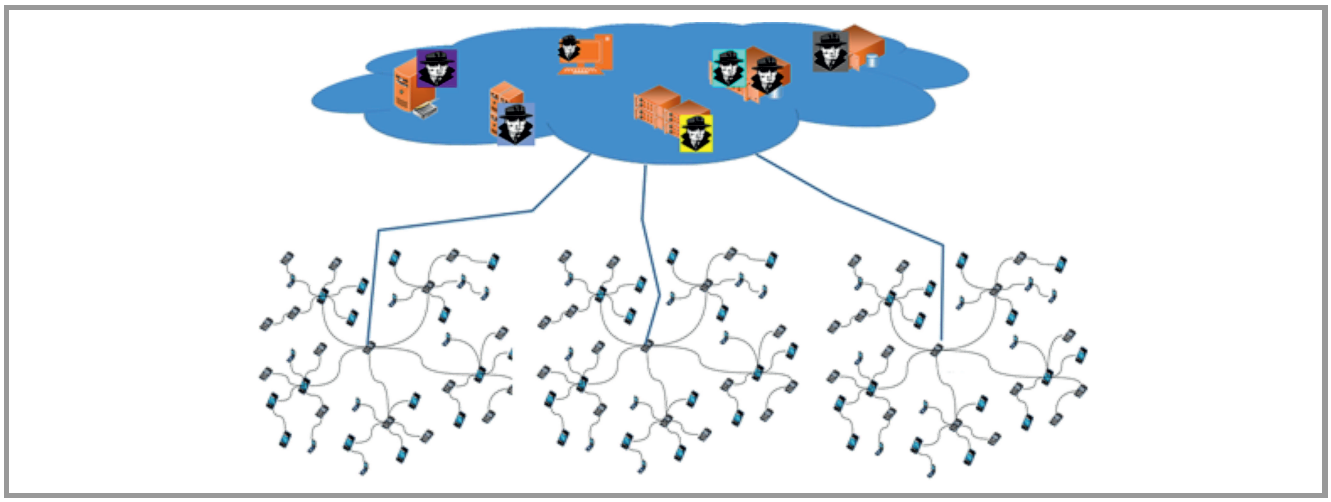

Fig. 17. The idea of cooperating mobile cloud and stationary computational environments.

is rapidly growing. Also, their computational power is greater and greater (quad-, hexa-, octa-core CPUs). Plus the same issues growing power regards memory, connectivity, battery, etc. So the answer can be simply "any". But the real advantage of the mobile devices used as the computational units is their personalization (your phone is always with you, and it is only yours) and localization in the real environment. So it seems that a particularly interesting task to be run on a cloud of the mobile devices would be a tasks related to environment sampling along with complex computations in the background, such as weather forecasting, air contamination monitoring and forecasting, distributed intelligent car navigation, and supporting public transportation system.

One of our further research and development directions is enhancing the presented mobile cloud with possibility of cooperating and spreading tasks to stationary computational units as presented in Fig. 17.

In this model, devices will be able to make decisions which tasks or which parts of particular tasks should be run on stationary computational environments (on the supercomputing center) and which of them should be run on the device. For instance, due to intensive cooperation with the end user, sampling and monitoring of the environment or intensive communication and cooperation among different devices and/or users are required to complete the given (part of) task.

Another very important issue is to work on the stability of the whole software configuration, therefore homogenization of the operating system will be considered (e.g. using Cyanogen Mod), tests will be rerun and extended, both for more hardware devices and for new problems.

Acknowledgements

This research was supported by AGH University of Science and Technology Statutory Fund.

References

[1] S. Mittal and J. S. Vetter, "A survey of CPU-GPU heterogeneous computing techniques", *ACM Computing Surv.*, vol. 47, pp. 69:1–69:35, 2015.

[2] U. Hansmann, L. Merk, M. S. Nicklous, and T. Stober, *Pervasive Computing: The Mobile World*. Springer, 2003.

[3] M. Satyanarayanan, "Mobile computing: The next decade", in *Proc. of the 1st ACM Worksh. on Mob. Cloud Comput. & Services: Social Networks and Beyond MCS'10*, San Francisco, CA, USA, 2010, pp. 5:1–5:6 (doi: 10.1145/1810931.1810936).

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility", *Future Gener. Comp. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[5] P. Mell and T. Grance, "The NIST definition of cloud computing", 2011 [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[6] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges", *IEEE Commun. Surv. & Tutor.*, vol. 16, no. 1, pp. 369–392, 2014.

[7] A. Khan, M. Othman, S. Madani, and S. Khan, "A survey of mobile cloud computing application models", *IEEE Commun. Surv. & Tutor.*, vol. 16, no. 1, pp. 393–413, 2014 (doi: 10.1109/SURV.2013.062613.00160).

[8] M. Rahimi, J. Ren, C. Liu, A. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions", *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133–143, 2014 (doi: 10.1007/s11036-013-0477-4).

[9] J. Samad, S. W. Loke, and K. Reed, "Mobile cloud computing", in *Cloud Services, Networking, and Management*, N. L. S. da Fonseca and R. Boutaba, Eds. Wiley, 2015, pp. 153–190 (doi: 10.1002/9781119042655).

[10] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models", *IEEE Commun. Surv. & Tutor*, vol. 16, pp. 393–413, 2014 (doi: 10.1109/SURV.2013.062613.00160).

[11] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey", *Future Gener. Comp. Syst.*, vol. 29, no. 1, pp. 84–106, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[12] A. Byrski, R. Dębski, and M. Kisiel-Dorohinicki, "Towards an agent-based augmented cloud", *J. of Telecom. and Inform. Technol.*, no. 1, pp. 16–22, 2012.

[13] A. Byrski, R. Dębski, and M. Kisiel-Dorohinicki, "Agent-based computing in an augmented cloud environment", *Int. J. of Comp. Syst. Science & Engin.*, vol. 27, no. 1, pp. 7–18, 2012.

[14] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, and S. Jeong, "Securing elastic applications on mobile devices for cloud computing", in *Proc. of the 2009 ACM Worksh. on Cloud Comput. Secur. CCSW'09*, Chicago, IL, USA, 2009, pp. 127–134.

[15] P. R. Elespuru, S. Shakya, and S. Mishra, "MapReduce system over heterogeneous mobile devices", in *Software Technologies for Embedded and Ubiquitous Systems*, S. Lee and P. Narasimhan, Eds. *LNCS*, vol. 5860, pp. 168–179. Berlin, Heidelberg: Springer, 2009.

[16] J. H. Christensen, "Using restful web-services and cloud computing to create next generation mobile applications", in *Proc. of the 24th ACM SIGPLAN Conf. Companion on Object Oriented Programm. Syst. Lang. and Appl. OOPSLA 2009*, Orlando, FL, USA, 2009, pp. 627–634.

[17] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing", *Mob. Netw. & Appl.*, vol. 16, no. 3, pp. 270–284, 2011 (doi: 10.1007/s11036-011-0305-7).

[18] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud", in *Proc. of the 6th Conf. on Comp. Syst. EuroSys 2011*, Salzburg, Austria, 2011, pp. 301–314.

[19] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing", *IEEE/ACM Trans. on Netw.*, vol. 24, no. 5, pp. 2795–2808, 2016 (doi: 10.1109/TNET.2015.2487344).

[20] E. Cuervo *et al.*, "Maui: Making smartphones last longer with code offload", in *Proc. of the 8th Int. Conf. on Mobile Systems, Appl., and Serv. MobiSys 2010*, San Francisco, CA, USA, 2010, pp. 49–62.

[21] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing", *J. of Netw. and Comp. Appl.*, vol. 59, no. C, pp. 46–54, 2016 (doi: 10.1016/j.jnca.2015.05.016).

[22] A. Botta, W. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey", *Future Gener. Comp. Syst.*, vol. 56, pp. 684–700, 2016 (doi: 10.1016/j.future.2015.09.021).

[23] M. Shiraz, A. Gani, R. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing", *IEEE Commun. Surv. Tutor.*, vol. 15, no. 3, pp. 1294–1313, 2013 (doi: 10.1109/SURV.2012.111412.00045).

[24] M. Gai, M. Qiu, L. Tao, and Y. Zhu, "Intrusion detection techniques for mobile cloud computing in heterogeneous 5G", *Secur. and Commun. Netw.*, vol. 9, no. 16, pp. 3049–3058, 2016 (doi: 10.1002/sec.1224).

[25] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, and K. Sakurai, "Authentication in mobile cloud computing: A survey", *J. of Netw. and Comp. Appl.*, vol. 61, pp. 59–80, 2016 (doi: 10.1016/j.jnca.2015.10.005).

[26] N. Kumar, K. Kaur, S. Misra, and R. Iqubal, "An intelligent RFID-enabled authentication scheme for healthcare applications in vehicular mobile cloud", *Peer-to-Peer Netw. and Appl.*, vol. 9, no. 5, pp. 824–840, 2016.

[27] C. Zhu, H. Wang, X. Liu, L. Shu, L. Yang, and V. Leung, "A novel sensory data processing framework to integrate sensor networks with mobile cloud", *IEEE Syst. J.*, vol. 10, no. 3, pp. 1125–1136, 2016.

[28] S. Murugesan and G. R. Gangadharan, Eds., *Harnessing Green IT: Principles and Practices*. Wiley, 2012.

[29] I. Lunden, "6.1B Smartphone Users Globally By 2020, Overtaking Basic Fixed Phone Subscriptions", Tecdhrunch, Jun. 2, 2015 [Online]. Available: https://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions

[30] 48th edition of the TOP500, nov. 2016 [Online]. Available: https://www.top500.org/lists/2016/11/

[31] H. Ba, W. Heinzelman, C. A. Janssen, and J. Shi, "Mobile computing – a green computing resource", in *Proc. 2013 IEEE Wirel. Commun. and Netw. Conf. WCNC 2013*, Shanghai, China, 2013, pp. 4451–4456.

[32] "T-mobile IPV6 is here and now", Tech. Rep., T-Mobile, 2013–2016 [Online]. Available: https://sites.google.com/site/tmoipv6/lg-mytouch

[33] N. Hemsoth, "Full details uncovered on chinese top supercomputer", HPC Wire, 2013 [Online]. Available: https://www.hpcwire.com/2013/06/02/full_details_uncovered_on_chinese_top_supercomputer/ (accessed 2016-12-15).

[34] M. Feldman, "China tops supercomputer rankings with new 93-petaflop machine", Top500, 2016 [Online]. Available: https://www.top500.org/news/china-tops-supercomputer-rankings-with-new-93-petaflop-machine/ (accessed 2016-12-15).

[35] T. Florin, "10 of the best octa-core smarphones available now", Aug. 17, 2014 [Online]. Available: http://www.phonearena.com/news/10-of-the-best-octa-core-smartphones-available-now_id59431

[36] H. Bauer, Y. Goh, J. Park, S. Schink, and C. Thomas, "The supercomputer in your pocket", *McKinsley on Semiconductros*, no. 2, pp. 14–27, 2012 [Online]. Available: http://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/semiconductors/issue%202%20autumn%202012/pdfs/the_supercomputer_in_your_pocket.ashx

Leszek Siwik obtained his Ph.D. in 2009 and works as an Assistant Professor in AGH University of Science and Technology. He is mostly interested in metaheuristics and multi-criteria optimization but also software engineering and mobile systems.

E-mail: siwik@agh.edu.pl
Department of Computer Science
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Krakow, Poland

Dawid Kala obtained his M.Sc. in 2016 from AGH University of Science and Technology. He is interested in software engineering and mobile systems.

E-mail: dawidkala@gmail.com
Department of Computer Science
AGH University of Science and Technology
Mickiewicza av. 30
30-059 Krakow, Poland

Mateusz Godzik obtained his M.Sc. in 2016 from Pedagogical University of Krakow and is currently a Ph.D. student at AGH University of Science and Technology. His interests cover cryptography, security and software engineering.

E-mail: godzik@agh.edu.pl
Department of Computer Science
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Krakow, Poland

Aleksander Byrski obtained his Ph.D. in 2007 and D.Sc. in 2013. He works as an Assistant Professor in AGH University of Science and Technology. He is interested in metaheuristics, agent-based systems, parallel and distributed computing.

E-mail: olekb@agh.edu.pl
Department of Computer Science
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Krakow, Poland

Wojciech Turek obtained his Ph.D. in 2010 and works as an Assistant Professor in AGH University of Science and Technology. He is interested in mobile robots, parallel and distributed computing and simulation.

E-mail: wojciech.turek@agh.edu.pl
Department of Computer Science
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Krakow, Poland

Marek Kisiel-Dorohinicki obtained his Ph.D. in 2001 and D.Sc. in 2013. He works as an Assistant Professor in AGH University of Science and Technology. He is interested in software engineering and agent-based systems.

E-mail: doroh@agh.edu.pl
Department of Computer Science
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Krakow, Poland