Some Capabilities of Android OS for Distributed Computing

Gergana Mateeva
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
gergana.mateeva@iict.bas.bg

Petar Tomov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
petar.tomov@iict.bas.bg

Dimitar Parvanov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
dimitar.parvanov@iict.bas.bg

Plamen Petrov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
plamen.petrov@iict.bas.bg

Georgi Kostadinov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
georgi.kostadinov@iict.bas.bg

Todor Balabanov[0000-0003-3139-069X]
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
todor.balabanov@iict.bas.bg

***Abstract*—Most of the algorithms in Computer Science are linear algorithms. This means that each subsequent instruction directly depends on the calculations of the previous instructions. In this type of linear algorithm, the sequence of calculations does not allow instructions to be calculated simultaneously on different processors. A much smaller subset of algorithms in Computer Science allows individual pieces of data to be processed in parallel. In turn, a subset of parallel algorithms allows implementation in the form of distributed computing. A very bright example of this is the population heuristics for global optimization and machine learning algorithms. Modern mobile devices, and especially Android OS based, are giving unlimited capabilities for distributed computing. In this study, some of these capabilities are discussed.**

***Index Terms*—Android OS, distributed computing, machine learning**

I. Introduction

Volunteer computing [4] is a branch of distributed computing [16]. In this variety of distributed computing, users donate their computing power at a time when their devices are not being used intensively. Time-consuming calculations [18] are performed on user devices, and the server only serves to synchronize and collect the results [6]. With such an organization of the calculation process, extremely high economic efficiency is achieved [10]. On the other hand, the reliability of the obtained results should always be questioned [13], as the calculations themselves are performed on devices that are not under the control of the organizer of the calculations.

978-1-6654-1042-7/21/$31.00 ©2021 IEEE

The beginning of the donated distributed computing was laid with the development of network communications and their transformation into an affordable service for every household. This development comes in combination with small software modules called screensavers [15]. For some time, desktops have worked with monitors based on a cathode ray tube. Static picture, in periods of the computer non-use, on this type of monitor had has led to damage to the thin layer of phosphor [19]. Activating the screensaver and its operating mode gives a clear indication that the computer is not in use [5]. All of the above leads to the fact that the first projects with donated distributed computing are implemented as screensavers. Nowadays, cathode ray tube monitors have not been used for a long time, but screensaver programs are still in use, at least to lock the work screen.

With the advent of smart mobile devices (phones and tablets), a series of new possibilities for donated distributed computing is opening up [21]. The main advantage of smart mobile devices is that they usually work around the clock. Their main disadvantage is that their processors are relatively low-power due to the need for smaller sizes and lower power consumption [3]. In addition to screensavers, smart mobile devices have active wallpapers, widgets, and daemon services. These additional software tools provide a wide range of possibilities for the implementation of distributed calculations.

This study gives a brief overview of some of the capabilities for distributed computing implementation in Android OS. The main contribution of the authors is a hybrid training of artificial neural networks. The process consists of Resilient Back-

propagation, Differential Evolution, and Genetic Algorithms [2]. After the introductory section, the paper is organized as follows: Section 2 introduces some common Android OS APIs; Section 3 presents a practical implementation and the usage of some third-party libraries; Section 4 concludes and provides some guidance for further research.

II. Android OS APIs

Android Live Wallpaper technology provides some of the most interesting opportunities for the implementation of distributed calculations. Live Wallpapers are interactive backgrounds with animation capabilities for the phone home screen. A live wallpaper is like other applications and can use almost of the same functionality [22]. A live wallpaper usually has dynamic content (hence the word "live"). Basic programming classes (WallpaperService and WallpaperService.Engine) do not provide any special functionality to performe an animation loop. It is the developer's responsibility to activate the animation loop, handle the life cycle events, suspend and resume the animations and deal with each animation step [9]. Live Wallpaper technology can be very helpful in the intermediate visualization of the results of distributed computations. At the same time, the Service is limited in terms of the time interval for activating the calculations.

Android Services is another very powerful tool for performing background calculations. A Service is an OS component used to perform time-consuming operations in the background. The Service does not provide a user interface as it is with the Activity. Once started, a Service might continue running for some time, even after the user switches to another Activity. Additionally, a component can bind to a Service and interact with it or perform inter-process communication. For example, a Service can deal with network transactions, perform file input/output, play music, or interact with a content provider, all over from the background [7]. Using Live Wallpaper only for intermediate visualization, the calculations themselves are displayed in a separate Services component.

Android Widgets is a component of the graphical user interface that can contribute with additional flexibility in the intermediate visualization of results. Widgets are an essential feature of mobile device screen organization. They can be presented as "at-a-glance" views of an application's most important information and functionality that is accessible right from the user's main screen. Users can move widgets across their screen panels, and resize them to tailor the amount of information within a widget, if resizing is supported by the widget [8]. Only chosen information calculated on the device is possible to be displayed with a gadget. This gives the user more freedom to organize the screen space and choose only the most important things. Widgets provide an additional option when distributed computing must involve user interaction. Interaction of this kind is common when there is a need for subjective human opinions such as an assessment of beauty or intuition. Widgets give much better interactivity than Live Wallpapers.

III. Practical Implementation

The described capabilities of Android OS are used in the mobile client application of the Vitosha Trade project [2]. The project is a joint development between Velbazhd Software LLC and the Institute of the Information and Communication Technologies at the Bulgarian Academy of Sciences. The client-side is an Android application used for training artificial neural networks. Backpropagation is one of the training algorithms. The other one is a Differential Evolution. The goal of the artificial neural network is financial time series forecasting.

Android OS APIs do not come with software libraries for machine learning. That is why third-party packages are used. For the multilayer perceptron, Java-based implementation of the Encog Machine Learning Framework is used (Listing 1). Encog is a Java/C# machine learning framework. It has been created to support genetic programming and different neural network technologies. Encog continues to be developed and bugs fixed. It includes types of models not covered by the large frameworks. A pure, non-GPU, Java/C# implementation is provided for several classic neural networks. Because it is written in Java, the source code of Encog can be much simpler for adaption in cases where someone wants to implement and ANN himself, from a scratch. The framework supports a diversity of advanced algorithms and support classes for data normalization and data preprocessing. Machine learning algorithms such as Hidden Markov Models, Genetic Programming, Genetic Algorithms, Support Vector Machines, Neural Networks, and Bayesian Networks are also supported. Most training algorithms are multi-threaded and scale well to multiprocessors hardware [12].

Listing 1. Building artificial neural network

```
network.addLayer(new BasicLayer(null, true, inputSize));
network.addLayer(new BasicLayer(new ActivationTANH(), true, hiddenSize));
network.addLayer(new BasicLayer(new ActivationTANH(), false, outputSize));
network.getStructure().finalizeStructure();
network.reset();
```

For the metaheuristic global optimization, Java-based implementation of the MOEA Framework is used (Listing 2). Single-objective implementation of Differential Evolution is involved in the application development. The MOEA Framework is a free and open-source, Java-based programming library. It has been created to develop and experiment with multiobjective evolutionary algorithms. It has usage in general-purpose single and multiobjective optimization. The framework supports differential evolution, genetic algorithms, grammatical evolution, genetic programming, particle swarm optimization, and more. A number of algorithms are provided out-of-the-box, including ε-MOEA, NSGA-II, NSGA-III, PAES, PESA2, SPEA2, GDE3, SMS-EMOA, CMA-ES, MOEA/D, IBEA, SMPSO, and OMOPSO. In addition, it provides the tools necessary for rapid design, development, execution, and statistical testing of optimization algorithms [11].

Listing 2. Differential Evolution training

```
Problem problem = new AnnErrorMinimizationProblem(weights, network, train);

List<Solution> solutions = new ArrayList<Solution>();
for (int i = 0; i < populationSize; i++) {
```

```
    Solution solution = problem.newSolution();
    solutions.add(solution);
}

AggregateObjectiveComparator comparator = new LinearDominanceComparator();
Initialization initialization = new InjectedInitialization(problem,
    populationSize, solutions);
DifferentialEvolutionSelection selection = new DifferentialEvolutionSelection();
DifferentialEvolutionVariation variation = new DifferentialEvolutionVariation(
    crossoverRate, scalingFactor);

AbstractAlgorithm algorithm = new DifferentialEvolution(problem, comparator,
    initialization, selection, variation);

long stop = System.currentTimeMillis() + optimizationTimeout;
while (System.currentTimeMillis() < stop) {
    algorithm.step();
}

weights = new ArrayList<Double>();
NondominatedPopulation population = algorithm.getResult();
if (population.size() > 0) {
    for (Double value : EncodingUtils.getReal(population.get(0))) {
        weights.add(value);
    }
}
```

Differential Evolution training is combined with backpropagation training (Listing 3). A problem with the backpropagation training is that the magnitude of the partial derivatives is usually larger or too smaller. Further, the learning rate is a single value for the whole artificial neural network. The resilient propagation algorithm uses an update value(similar to the learning rate) for every artificial neuron connection. The bigger advantage is that update values are automatically calculated, which is not the case with the learning rate in the classical backpropagation training. In most machine learning situations, resilient propagation is suggested for training instead of classical backpropagation. There are three parameters that must be provided to the resilient training. Default values are suggested for each, and in almost all cases, the default values are absolutely sufficient. All simplifications make the resilient propagation training an easier one and the most efficient training algorithms available [12].

Listing 3. Resilient backpropagation training

```
train = new ResilientPropagation(network, examples);
train.iteration();
train.finishTraining();
```

Artificial neural network structure and financial times series are taken on a regular basis from the lightweight PHP/MySQL server. Communication is done via HTTP protocol (Listing 4) with third-party HttpClient Android Library [20]. Google's Android OS 1.0 was released with a pre-beta snapshot of Apache HttpClient. To coincide with the first Android release, Apache HttpClient 4.0 APIs had to be frozen prematurely. Many of its interfaces and internal structures were still not fully worked out. As Apache HttpClient 4.0 was maturing, it was expected Google to incorporate the latest code improvements into their code tree. Unfortunately, such incorporation did not happen at all. A version of Apache HttpClient shipped with Android has effectively become a fork [1].

Listing 4. HTTP communication

```
HttpClient client = new DefaultHttpClient();
client.getParams().setParameter("http.protocol.content-charset", "UTF-8");

HttpPost post = new HttpPost("http://" + url.trim() + "/" + LOAD_RANDOM_ANN_SCRIPT);
HttpResponse response = client.execute(post);

HttpPost post = new HttpPost("http://" + url.trim() + "/" + LOAD_TRAINING_SET_SCRIPT);
post.setEntity(new UrlEncodedFormEntity(pairs));
HttpResponse response = client.execute(post);

HttpPost post = new HttpPost("http://" + url.trim() + "/" + SAVE_RETRAINED_ANN_SCRIPT);
post.setEntity(new UrlEncodedFormEntity(pairs));
client.execute(post);
```

For a structural exchange of the information, a JSON-based communication protocol is used. All messages are packaged as JSON data or key/value pairs (Listing 5). For JSON messages handling, a third-party library JSON-java is used. JSON is a light-weight language-independent data interchange format. The JSON-Java package is a reference implementation that demonstrates how to parse JSON documents into Java objects and how to generate new JSON documents from the Java classes [17].

Listing 5. JSON packaging

```
JSONObject result = new JSONObject(EntityUtils.toString(response.getEntity(), "UTF-8"));

int size = result.getInt(JSON_SIZE_KEY);
symbol = result.getString(JSON_SYMBOL_KEY);
period = result.getInt(JSON_PERIOD_KEY);
double fitness = result.getDouble(JSON_FITNESS_KEY);
int numberOfNeurons = result.getInt(JSON_NUMBER_OF_NEURONS_KEY);
JSONArray flags = result.getJSONArray(JSON_FLAGS_KEY);
JSONArray weights = result.getJSONArray(JSON_WEIGHTS_KEY);
JSONArray activities = result.getJSONArray(JSON_ACTIVITIES_KEY);

int size = result.getInt(JSON_NUMBER_OF_EXAMPLES_KEY);
JSONArray time = result.getJSONArray(JSON_TIME_KEY);
JSONArray open = result.getJSONArray(JSON_OPEN_KEY);
JSONArray low = result.getJSONArray(JSON_LOW_KEY);
JSONArray high = result.getJSONArray(JSON_HIGH_KEY);
JSONArray close = result.getJSONArray(JSON_CLOSE_KEY);
JSONArray volume = result.getJSONArray(JSON_VOLUME_KEY);
```

A single cycle of artificial neural network training and forecasting is done within a separate thread of the Live Wallpaper service (Listing 6). A separate thread is needed because Android OS is very sensitive to long-running calculations. If long-running calculations are performed in the thread of the graphical user interface, this will block the entire application and slow down the overall responsiveness of the operating system. The Android OS has a very strict policy related to blocking its graphical user interface. Such applications are shut down very quickly.

Listing 6. Calculations in separate thread

```
private final Runnable trainer = new Runnable() {
    public void run() {
        predictor.predict();
        draw();
        predictor.train();
    }
};
```

It is a well-established practice in volunteer distributed computing some intermediate results to be visualized during the calculation process (Listing 7). In the Vitosha Trade Android Client project, the photos of Vitosha Mountain are shown as a background (Fig. 1). Over the background, three separate panels are visualizing calculating progress information (Listing 7).

Listing 7. Intermediate results visualization

```
SurfaceHolder holder = getSurfaceHolder();
Canvas canvas = null;

try {
        canvas = holder.lockCanvas();

        if (canvas != null) {
                drawBackground(canvas);
                drawPanels(canvas);
                drawCurrencyPairInfo(canvas);
                drawForecast(canvas);
                drawAnn(canvas);
        }
} finally {
        if (canvas != null) {
```

```
                holder.unlockCanvasAndPost(canvas);
            }
        }

        handler.removeCallbacks(trainer);

        if (visible == true) {
            handler.postDelayed(trainer, delay);
        }
```

The capabilities for the alpha channel inside the Canvas object are used for semi-transparent visualization. The first panel shows the time series financial thicker (Bitcoin vs USD) and the measurements interval (daily). The second panel visualizes past values as bars in green and visualizes forecasted values as bars in red. The third panel visualizes a multilayer perceptron. On the left side of the panel, the values of the input layer are shown in green. After input values, the values of the weights between input and hidden layer are shown in grayscale. The values of the hidden layer are shown in blue in the center of the panel. Next to the hidden layer values are shown the values of the weights between the hidden and the output layer, again in grayscale. On the right side of the panel, the values of the output layer are shown in red. The length of the five columns is different, so scaling is done to fit the visualization into the rectangular shape of the panel.

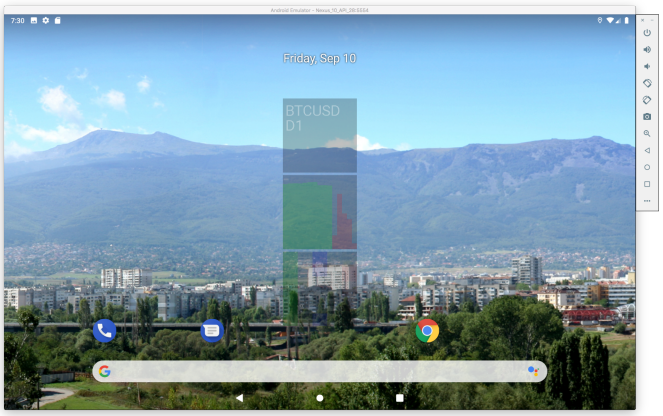

Fig. 1. Android Live Wallpaper distributed computing

The wallpaper has the following settings (Fig. 2) - the URL address of the remote server (Fig. 3), device loading (Fig. 4), positions of the panels (Fig. 5), and size of the panels (Fig. 6).

The mobile client application is written in such a way that the calculations can be performed even when the mobile device is offline on the Internet. This is achieved by storing the calculation information in a local copy of the SQLite relational database. If it is not possible to request a calculation package from the remote server, a package from the local database is loaded.

In some projects for distributed computing, it is impossible to calculate the quality of the obtained solutions numerically. Examples of such projects are situations in which a subjective assessment of beauty is needed. Another situation is subjective human intuition. Many people make many decisions based on their intuitive opinion. It is popular to do stock exchange

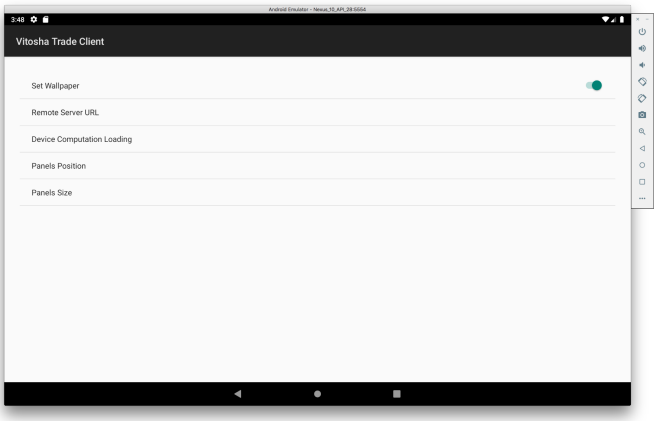

Fig. 2. Wallpaper performance settings

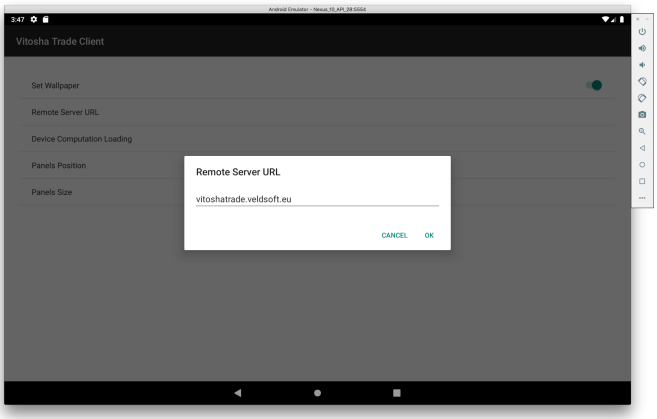

Fig. 3. Remote server URL address

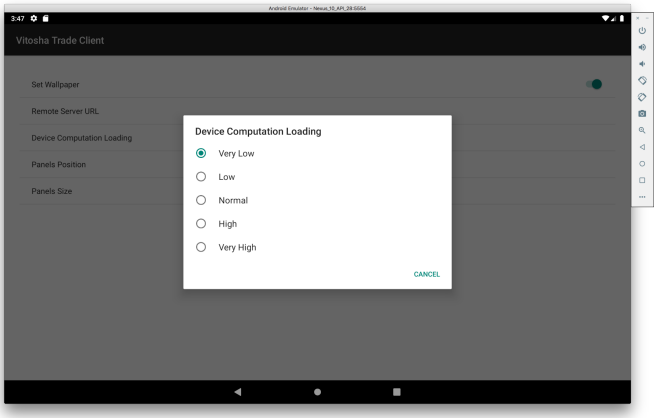

Fig. 4. Device background calculations loading

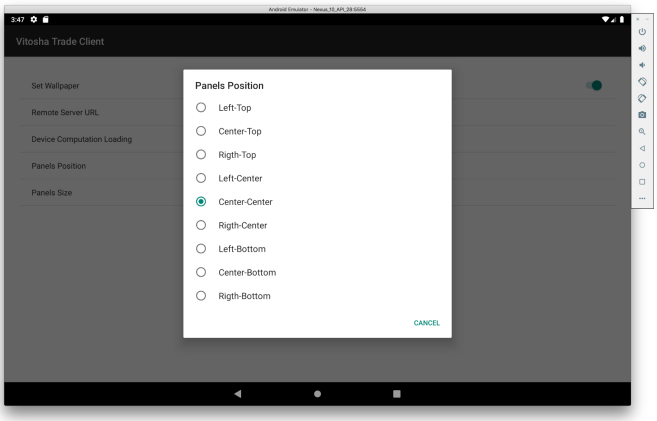

Fig. 5. Position of the panels

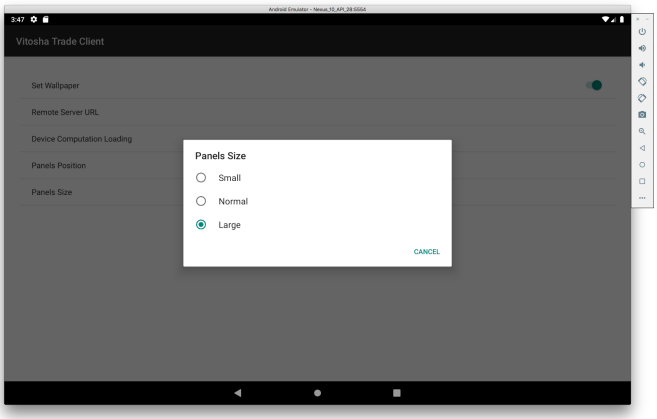

Fig. 6. Size of the panels

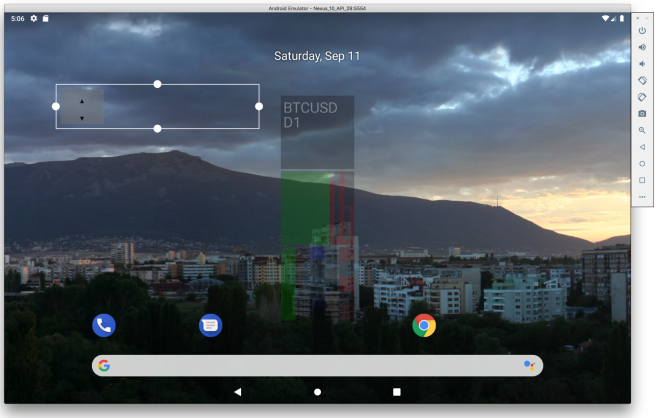

Fig. 7. A voting widget

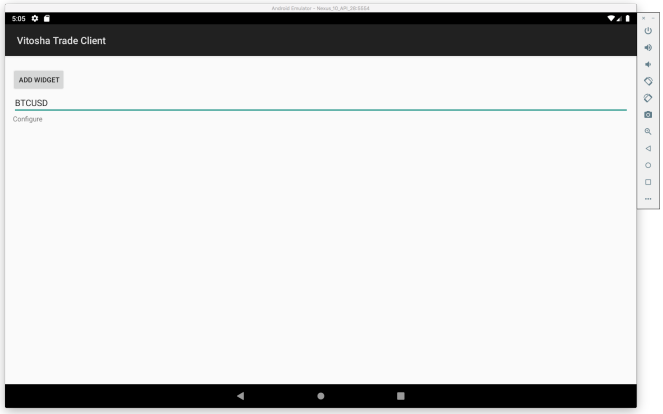

Fig. 8. Settings of the voting widget

trading, done only because of the feeling that a certain company will perform well. Decisions made by people only through intuition are difficult to explain logically. The fact that decisions are difficult to explain does not mean that these decisions cannot be correct in most cases. The intuitive solution is the result of a large amount of information that people receive on a daily basis and is very often processed mainly on a subconscious level. Such information processing is not possible with modern computers. The information, processed in such a way, can be collected in human-computer distributed computing. In a financial forecasting system, the user can be asked to vote for prices up or down. Android Widgets are the perfect tool for gathering such voting information (Fig. 7). Widgets are not only static visualizers, but they can handle user input. The votes are first stored in the local SQLite database and then sent to the remote server. The important information is which financial instrument is being voted on, when the voting took place in time and what is the voting direction. The voting information is classified on the server-side according to the frequency of voting and the success of the voting assumption [14]. The voting widget has a settings dialog itself (Fig. 8).

Because Android OS allows external libraries written in Java, it gives almost unlimited possibilities for machine learning. The modular development of the project allows calculations to be done even on desktop machines by recompilation of the client source code [2]. Such organization of the source code allows mobile devices to work in parallel with desktop computers. Different metaheuristics can be involved in artificial neural networks training. By its communications and user interface capabilities, Android OS is a perfect candidate for distributed computing projects. The rapid development of mobile hardware will offer great computing power in the near future.

IV. Conclusion

This study presents the capabilities of the Android OS for the implementation of donated distributed computing. The capabilities of the operating system are demonstrated with an

application for training artificial neural networks. The training has been done with population-based heuristics for global optimization. The artificial neural network has been trained to forecast time series. The efficiency of mobile distributed computing has been proven, but testing of the system is still in progress.

Having all these distributed computing capabilities in Android OS is inspiring, but as further research, it will be interesting what can be done with iOS devices and Kai OS devices.

ACKNOWLEDGMENT

This research is funded by Velbazhd Software LLC and it is partially supported by the Bulgarian Ministry of Education and Science (contract D01–205/23.11.2018) under the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", approved by DCM # 577/17.08.2018.

REFERENCES

[1] Apache HttpClient from Android, The Apache Software Foundation, 2021. https://hc.apache.org/httpcomponents-client-4.5.x/android.html
[2] T. Balabanov, Vitosha Trade Android Client, Velbazhd Software LLC, 2021. https://github.com/TodorBalabanov/Vitosha-Trade-Android-Client
[3] S. Chinara, S. Rath, Energy Efficient Mobility Adaptive Distributed Clustering Algorithm for Mobile Ad Hoc Network, Proceedings of International Conference on Advanced Computing and Communications, pp. 265-272, 2008.
[4] M. Durrani, A. Shamsi, Volunteer computing: requirements, challenges, and solutions, Journal of Network and Computer Applications, vol. 39, pp. 369-380, 2014.
[5] K. Forcier, Never Idle: The Animated Screensaver and the Culture of Always-On Computing, Afterimage, vol. 48, no. 2, pp. 79–93, 2021.
[6] J. Geng, D. Li, Y. Cheng, S. Wang, J. Li., HiPS: Hierarchical Parameter Synchronization in Large-Scale Distributed Machine Learning, Proceedings of the 2018 Workshop on Network Meets AI & ML, Association for Computing Machinery, New York, NY, USA, pp. 1-7, 2018.
[7] Google Developers, Services Overview, Google LLC, 2021. https://developer.android.com/guide/components/services
[8] Google Developers, App Widgets Overview, Google LLC, 2021. https://developer.android.com/guide/topics/appwidgets/overview
[9] K. Grouchnikov, Live wallpapers with Android SDK 2.1, Pushing Pixels, 2010. https://www.pushing-pixels.org/2010/02/01/live-wallpapers-with-android-sdk-2-1.html
[10] H. Guler, B. Cambazoglu, O. Ozkasap, Task allocation in volunteer computing networks under monetary budget constraints, Peer-to-Peer Networking and Applications, vol. 8, pp. 938-951, 2015.
[11] D. Hadka, MOEA Framework, A Free and Open Source Java Framework for Multiobjective Optimization, http://moeaframework.org/
[12] J. Heaton, Encog Machine Learning Framework, Heaton Research Inc., 2020. https://www.heatonresearch.com/encog/
[13] E. Heien, D. Anderson, K. Hagihara, Computing Low Latency Batches with Unreliable Workers in Volunteer Computing Environments, Journal of Grid Computing, vol. 7, pp. 501, 2009.
[14] R. Ketipov, G. Kostadinov, P. Petrov, I. Zankinski, T. Balabanov, Human-Computer Mobile Distributed Computing for Time Series Forecasting, Proceedings of Distributed Computer and Communication Networks, Communications in Computer and Information Science, Springer, Cham, vol. 1141, pp. 503-509, 2019.
[15] E. Krieger, G. Vriend, Models@Home: distributed computing in bioinformatics using a screensaver based approach, Bioinformatics, vol. 18, no. 2, pp. 315-318, 2002.
[16] L. Lamport, N. Lynch, Distributed Computing: Models and Methods, Handbook of Theoretical Computer Science, Formal Models and Semantics, pp. 1157-1199, 1990.
[17] S. Leary, JSON in Java [package org.json], 2021. https://github.com/stleary/JSON-java
[18] I. Meyerov, S. Bastrakov, A. Sysoyev, V. Gergel, Comprehensive Collection of Time-Consuming Problems for Intensive Training on High Performance Computing, Communications in Computer and Information Science, vol. 965, Springer, Cham, pp. 523-530, 2019.
[19] A. Pfahnl, Properties of fast-decay cathode-ray tube phosphors, The Bell System Technical Journal, vol. 42, no. 1, pp. 181-201, 1963.
[20] M. Sebera, HttpClient Android Library, 2019. https://github.com/smarek/httpclient-android
[21] C. Tapparello, C. Funai, S. Hijazi, A. Aquino, B. Karaoglu, H. Ba, J. Shi, W. Heinzelman, Volunteer Computing on Mobile Devices: State of the Art and Future Research Directions, Mobile Computing and Wireless Networks: Concepts, Methodologies, Tools, and Applications, pp. 2171-2198, 2016.
[22] L. Vogel, Android Live Wallpaper - Tutorial, Vogella GmbH, 2020. https://www.vogella.com/tutorials/AndroidLiveWallpaper/article.html