IOP Conf. Series: Materials Science and Engineering **327** (2018) 022070 doi:10.1088/1757-899X/327/2/022070

Distributed Information and Control system reliability enhancement by fog-computing concept application

E V Melnik[1], **A B Klimenko**[2], **D Ya Ivanov**[2]

[1] Southern Scientific Center of the Russian Academy of Science, 41 Chehova St., Rostov-on Don, 344006, Russia
[2] Scientific-Research Institute of Multiprocessor Computing Systems of Southeren Federal University, 2 Chehova St., Taganrog, 347928, Russia

E-mail: anna_klimenko@mail.ru

Abstract. The paper focuses on the information and control system reliability issue. Authors of the current paper propose a new complex approach of information and control system reliability enhancement by application of the computing concept elements. The approach proposed consists of a complex of optimization problems to be solved. These problems are: estimation of computational complexity, which can be shifted to the edge of the network and fog-layer, distribution of computations among the data processing elements and distribution of computations among the sensors. The problems as well as some simulated results and discussion are formulated and presented within this paper.

1. Introduction

Contemporary Information and Control Systems (ICS) are the integral components of a large amount of mechatronic complexes, such as: complexes in oil and gas production and refinery, hazardous industries, autonomous robotics, energy plants, aircraft, spacecraft, etc.

The good example of such systems is an oil-production. For the oil wells exploitation the complex mechatronic objects on the basis of electric centrifugal pumps (ECP) are used. The ECP downtime leads to the economic losses from 250000 to 1000000 rubles [1]. So, such systems require a dependability of a high level.

This paper focuses on the ICS reliability, which is the aspect of a system dependability. The issue of the contemporary ICSs is that the amount of the data to be processed is quite huge and leads to the poor scalability, high system latency and high loads of the information processing units within the computational environment.

A new computing paradigm – the fog-computing [2-4] – was synthesized to solve the problems mentioned above. As fog-computing is quite new, the one's application is not systemized and is on the development stage (examples of the fog-computing application are described in the following sections of this paper).

The clue idea of the current research is that the application of the fog-computing concept to the ICS functional architecture can enhance the reliability function of the system while issues of latency, scalability and communication overheads are removed too. On the current stage of the research, let us propose an approach of reliability enhancement based on the application of the fog-computing concept to ICS. The method proposed is a three-component one and includes a complex of multicriteria

optimization problems, which are formalized and presented within this paper.

2. Traditional ICS, its architecture and reliability

Traditionally, ICS can be presented by the following structure: sensors, computational environment (CE), actuators. The progress in the microelectronic and sensor device area allows one to provide such devices with a network interface. It leads to the possibility of integration of all types of sensors, data processing units and actuators into the network. So, some communication infrastructure is in usage too. The general scheme of the traditional ICS architecture is given in Figure 1.

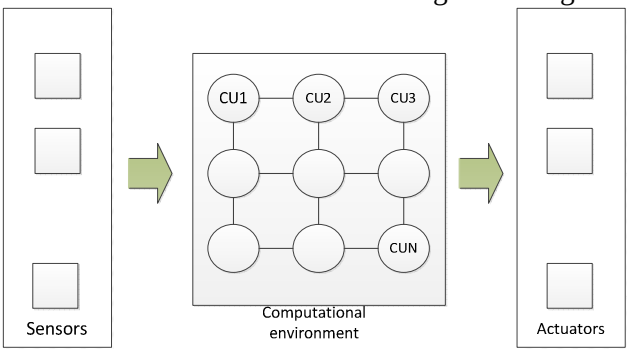

Figure 1. Traditional ICS architecture.

If sensors are located remotely from the computational environment, and amount of data to be processed is large, the following issues are of current interest [5]:

- system latency. The main reasons for this are the network hops and the large amount of data to be transferred to the computational environment;
- high computational environment load. It leads to the reliability function degrating;
- high communicational environment load. It leads to the reliability function degradation too;
- low system scalability.

Turning to the reliability function, on one hand, it merges reliabilities of all elements of the system, on the other hand, it depends on the element loadings.

The reliability function is given in equation (1):

$$P(t) = e^{-\lambda t}, \tag{1}$$

where where λ is a failure rate and a constant.

Failure rate depends on the temperature of the element, as is shown in equation [6] (2):

$$\lambda = \lambda_0 \cdot 2^{\Delta T/10} \tag{2}$$

So, assuming that temperature of the element is the ratio of the element load, the equation (2) can be presented as follows:

$$\lambda = \lambda_0 \cdot 2^{kD/10} \tag{3}$$

where k – is the ratio of the load-temperature dependency, D – the loading of the element.

Therefore, the reducing of the computational unit (CU) load within the computational environment leads to the computational units reliability function enhancement. The load reducing is reachable by the moving of some computations to the edge of the network and to the fog layer. Communication overheads reducing affects the CU loading positively too.

3. Fog-computing concept

The fog-computing concept is quite new and supposed to solve the latency, scalability and other problems of Internet of Things. Indeed, in the conditions of data exchange with cloud, it is expedient to delegate some computations to the nearest devices.

It is useful to mention that fog computing and edge computing are the different terms, although they are frequently confused [7].

Fog computing pushes intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway. Edge computing pushes the intelligence, processing power and communication capabilities of an edge gateway or appliance directly into devices like programmable automation controllers.

As it is impossible to fully delegate the process of decision making in ICSs to the edge of the network, the fog computing concept is applicable to delegate some preliminary data processing from the CE to the sensors and other network devices nearby. Similar idea was proposed in [8].

The obvious advantages of the fog computing concept usage are:

- system latency decreasing;
- CE load decreasing;
- communication network load decreasing;
- system scalability improving.

The general distinguishes between the fog-computing concept and the traditional, “cloud” architecture are shown in figure 2.

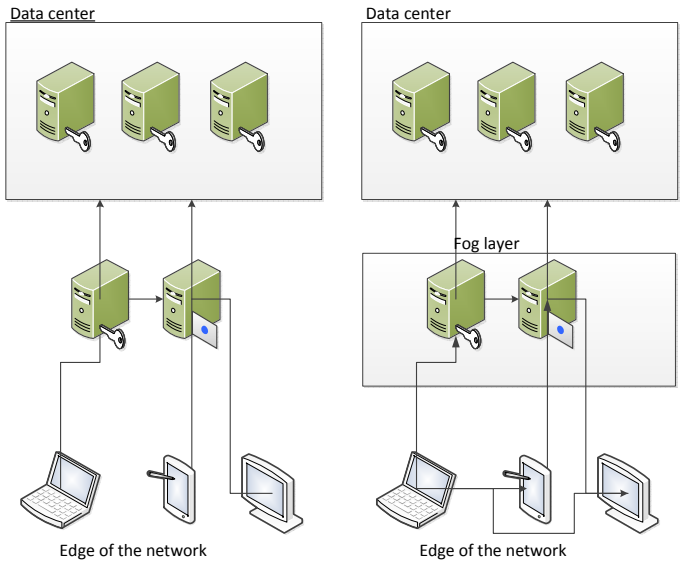

Figure 2. General distinguishes between cloud and fog computing.

While contemporary sensors and communication infrastructure devices have sufficient computational resources and network interfaces, distribution of the computations to the fog layer reduces the computational load of computational units (CUs), and so enhance its reliability functions.

4. Distributed Information and Control system reliability enhancement approach

Within this paper the new approach to the distributed ICS reliability enhancement by fog-computing concept usage is presented.

The approach is a complex one and includes three general steps:

1. Distributed computation planning. Too much of computational load, shifted to the edge of the network, can reduce the reliability functions of the network edge devices and so, as a result, overall reliability of the system will be of a poor quality. So, the first step – distributed computation planning

– is for some estimation, planning and developing computational tasks so as maximize the overall system reliability function. As reliability function model depends on a system particular implementation, here let us present just a casual diagram of term relationships (see figure 3).

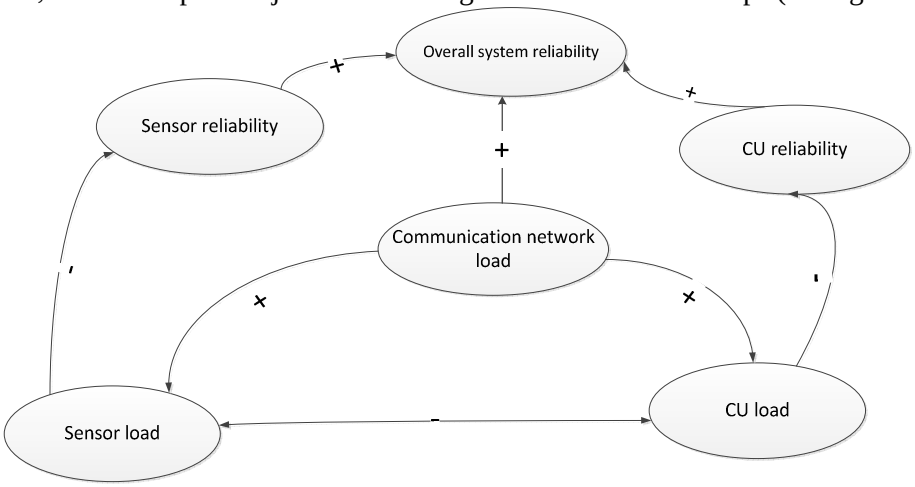

Figure 3. Casual diagram of general term relationships.

Applying to the particular ICS, the first step of approach presented presupposes the particular optimization problem solving with overall reliability function criterion. After this step, one has some planned computational tasks with determined computational complexity. So, a part of these processes can be shifted to the edge of the network with guarantee that the overall reliability function will not degrade. The other part of computational tasks can be distributed among the CUs within CE.

2. The load distribution among the CUs of CE

As is presented in [9-11], the model of configuration forming problem is suited for the current case of task distribution. Let us assume that the CE relates to the class of ICSs with performance redundancy [12]. So, the model of load distribution of computational tasks will be as follows.

There are N MCTs with computational complexities g_i, M CUs with equal performance m_j , $U=\{u_{ij}\}$ – the percentage of j CU performance allocated for the i MCT, T – planned completion time for the N MCTs, $F = \{f_k\}$, $k \in \{1,...M\}$, – the set of simultaneously failed CUs.

Through the resource allocation every MCT links to the CU, and it can be described by the following tuple:

$a_i=<j, u_{ij}, t_i>$, where j – the CU identificator, u_{ij} – the allocated resource ratio, t_i – the time of MCT i start.

So, the set $A=\{a_i\}$ determines the configuration of ICS before failure, the set $A'=\{a'_i\}$ determines the configuration of ICS after the reconfiguration. In fact, A' is the solution of configuration forming problem, and a'_i – the tuples which describe the new MCT assignments.

The objective functions are the following.

Firstly, the number of MCTs relocated from the operational nodes must be minimized. In other words, if there is a solution where the MCT's new assignment propose the relocation of tasks from the operational node, one should choose the solution, where there is no relocation at all. This objective function can be described with the expressions given below.

Let's determine the subtraction operator for sets A and A' so that:

$$a_i - a_i^{'} = \begin{cases} 0, if\ \ j\ is\ equal\ to\ \ j'; \\ 1, otherwise. \end{cases} \quad (4)$$

Then

$$F_1 = \sum_{i=1}^{N} (a_i - a_i^{'}) \rightarrow MIN \tag{5}$$

The optimal location in the search space of this objective function means that only MCTs from the faulted node are launched on other nodes.

The second objective function is the minimization of the eliminated MCTs. On practice some MCTs are critical and must be saved during the reconfiguration, and some MCTs are non-critical. But from the system survivability point of view it is extremely preferable to save as much MCTs as possible. So,

$$F_2 = |A| - |A'| \rightarrow MIN \tag{6}$$

And, finally, the dispersion of CU loadings must be minimized:

$$F_3 = \sum_{k=1}^{K} u'_{kj} - \sum_{l=1}^{L} u'_{lq} \rightarrow MIN, \ \forall j, q \tag{7}$$

where K is the number of MCTs assigned to the CU j, L is the number of MCTs assigned to the CU q.

The main constraint is that all MCTs must be accomplished within the planned completion time T:

$$t'_i + \frac{g_i}{u'_{ij} \cdot m_j} \le T, \ \forall i, j. \tag{8}$$

Also the failed CUs must be taken into consideration:

$$M' = M - F, \tag{9}$$

where M', M and F are the sets of CUs.

And, lastly, the bordering conditions are: all variable values are positive, $u_{ij}<1$, $u'_{ij}<1$, $\forall i, j$. So, the vector objective function contains objective functions (5)-(7), with constrains (8),(9).

3. Computations on the edge of the network: device community forming

To continue, some new terms must be defined.

Preliminary data processing device (**PDPD**) is a device, which can participate the preliminary data processing as a member of device community.

Preliminary data processing task (**PDPT**) is a set of operations, which must be performed before the pre-processed data is sent to the **CE**.

The device community forming (DCF) problem model bases on the following assumptions:

1. Let us presuppose some degree of uncertainty of the PDPD number, but if PDPD solves a task, it must finish it.

2. All PDPDs can exchange the information about their states.

Let's take a look on the preferable criteria for the optimization problem under consideration.

1. It is expedient to minimize the community cost. This criterion relates to the possibility and necessity to solve a large amount of PDPTs in a distributed manner on the edge of network.

2. As PDPT can be performed in a distributed manner, its structure can be described as an information graph. In such graph vertexes are the sub-operations, and ribs are the information flows between sub-operations. So, the next criterion is the amount of data exchange during the PDPT performing with the condition of sub-operation distribution through the device community.

3. The criterion of device load is due to the dependence between the load and reliability function [7]. The dependency is exponential, so, the load minimization is rather desirable component of the

general objective function.

The DCF problem can be preliminary formulated as follows: having PDPT parameters, planning PDPT completion time and PDPD identifier given, the device community and the PDPT subtasks performing schedule must be formed with completion time constraint and previously formulated criteria optimization.

The formal model of DCF problem is presented below.

A set of PDPDs is incorporated to the network of the arbitrary topology.

PDPD is characterized by the tuple given below:

$<j, p_j, E_j, \{R_{jk}\}, \{F_{jk}\}, u_j, s_j>$,

Where

j – PDPD identifier;

P_j – PDPD performance;

$\{R_{jk}\}$ – the list of distances between PDPDs. Here the distance is the number of transitory network segments.

$\{F_{jk}\}$ - the list of system PDPD computational resources. It must be refreshed periodically through the system operating stage;

u_j – available computational resource of the PDPD j;

s_j – the cost of PDPD j;

v – data transmission speed, in order to simplify the problem model, it is identical for all network segments.

The next step for our DCF problem model formalization is the description of the PDPTs.

The input data for a PDPT can be some variations of data, which must be pre-processed by the device community.

For instance, it can be an image batch, or a video batch, or huge amount of numbers, generated by sensor. Let us assume that the types of such data is predefined, so the software components can be constant too. The programs of the data preprocessing are described formally as a set of graphs, and each graph can be performed in a distributed manner.

Let PDPT i be described by a graph g_i, where the vertexes are the code fragments with computational complexity a_k, ribs are the data flows between code fragments, w_{dk} – the amount of data which must be transferred between code fragments k and d. Here the auxiliary variable must be defined: f_{dk}=1, if data are transferred through the network, and f_{dk}=0, if code fragments are performed on the same device.

The set of PDPTs to be preprocessed is $G=\{g_i\}$. Every PDPT has its t_i , time of the preprocessing beginning, and T_i , planned completion time.

The key idea of the fog-computing is the system latency minimization by the computation distribution among the neighbor devices. So, while device community forming, it is expedient to choose devices with minimum distance between them.

It must be noted that at moment t_i PDPD j can have insufficient computational resources u_{ij}. This computational resource can be insufficient to solve any code fragment a_k of the PDPT i, and so the task must be given to another PDPDs.

There are two main ways of such situation development:

- j-identifier is connected to the PDPT description and the PDPT result. It is translated to the CE after the PDPT solving. In this case PDPT can migrate everywhere from the initial device, but, obviously, the community must not be too far. If it is so, all pros of the fog computing concept usage are eliminated.
- If PDPD has not got enough computational resources, it can give its PDPT to the community, but always waits the result and send it to the CE itself. In this case the distance between initial PDPD and community formed must be as short as possible too, as was mentioned in previous case.

Generalizing, let us presuppose that in all cases it is expedient to form the device community somewhere nearby the initial PDPD. So the authors formed the additional optimization criterion: the

distance between the community participants.

Situations, presented above, illustrated in figure 4.

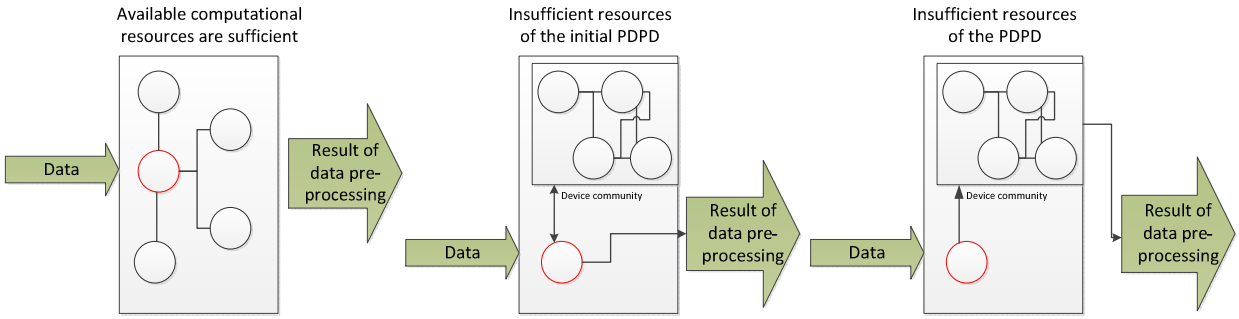

Figure 4. The community forming with insufficient resources and task of PDPT migration

The identifier of the initial PDPD, which must preprocess PDPT, $J_{cur.}$

Then, the formal description of the criterion of the distance between community devices can be presented as follows:

$$F_1 = \max_k R_{J_{cur},k} \rightarrow \min \tag{10}$$

The criterion of the data flows between code fragments, transferred through the network, can be presented as:

$$F_2 = \sum_{k=1}^{M} w_{dk} \cdot f_{dk} \rightarrow \min, \tag{11}$$

where M is the number of devices in community.

3. Community cost criterion:

$$F_3 = \sum_{j=1}^{M} s_j \rightarrow \min, \tag{12}$$

where s_j is the cost of PDPD j from the community.

4. PDPD load intense:

$$F_4 = (1-u_j) \rightarrow \min, \forall j \subseteq M. \tag{13}$$

The first constraint is a completion time constraint:

$$\forall k, j \ \max(t_i + \frac{a_k}{p_j u_j}) \le T_i \tag{14}$$

Besides this, every subtask a_k cannot be solved earlier than its predecessor in the graph g_i:

$$\forall k, j, t_i + \frac{a_l}{p_j u_j} + \frac{w_{lk}}{v} < t_i + \frac{a_k}{p_j u_j} \tag{15}$$

As the available resources are known in the system, each computational resource related to the PDPD takes one subtask a_k to perform.

So, the final DCF problem can be formulated as following: with given PDPT parameters g_i, PDPT planned time completion T_i, PDPD with initial data identifier j and available computational resources u_j, the subset of PDPD and subtask a_k of g_i distribution among the PDPD community must be formed

so that constraints (14, 15) are satisfied and the vector objective function $F=(F_1, F_2, F_3, F_4)$(10-13) is minimized.

5. Discussion and simulation results

According to presented approach, the first step is to estimate how much of computational complexity can be shifted from the CE to the edge of the network and the fog layer. The question is quite complicated, because of need of adequate model of system reliability.

The simplest simulation allows one to estimate the dependencies of CU and sensor reliability function vaues: assuming the system without failures, one can estimate the system overall reliability, as is shown in figure 5.

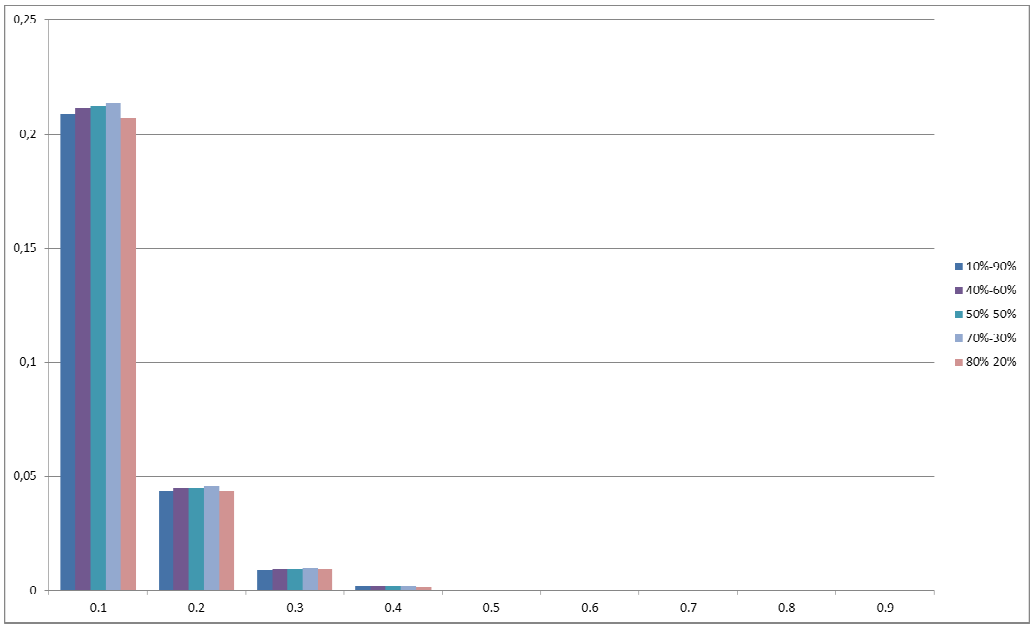

Figure 5. The overall reliability of the system

According to the results of the simulation, the worst overall reliability function value is for the 90% loading of the sensors (10 units for the simulation), and 10% loading of the data processing CUs (5 units for the simulation), and for 80% loading of CUs and 20% loading of sensors. It is seen that there is an optimum point for the computation distribution among the CE and fog layer.

As to other two steps and two optimization problems, the solutions can be reached by the techniques, presented in previous works [9-12]. Briefly, the multicriteria problem is transferred to the problem of one criterion, and the solutions are got with the simulated annealing algorithms.

6. Conclusions.

The current paper focuses on the ICS reliability issue. The new complex approach is presented, in which the positive effect is reachable by the fog-computing concept usage.

Within this paper, main stages of the approach presented are formulated and solved particularly via optimization problems.

The first stage model is formulated in general terms, other two stages are formalized as multicriteria optimization problems. Simulation results underline the complexity of the first approach stage, while other two optimization problems were solved in related papers.

Resuming, it is expedient to mention that the problem of ICS reliability is one of the most vital in the contemporary world. Applying the fog-computing concept elements, the reliability function of the

data processing units can be improved. It makes this approach quite promising and topical for the future research.

7. Acknowledgments

The reported study was funded by RFBR No.17-08-01605 and by SSC RAS project 02562014-0008 within task 007-01114-16 PR.

References

[1] Korovin I S 2016 *Izvestiya UFU. Technicheskiye nauki* **71(16)** 59-64
[2] Yi S, Hao Z, Qin Z and Li Q 2016 *HotWeb 2015* 73–8
[3] More P 2015 *Int. J. Res. Eng. Technol.* **4** 2319–22
[4] Stojmenovic I and Wen S 2014 *Proc. 2014 Fed. Conf. Comput. Sci. Inf. Syst.* **2** 1–8
[5] Hosseinpour F, Meng Y, Westerlund T, Plosila J, Liu R and Tenhunen H 2016 *Inretnational J. pf Adv. Comput. Technol.* **12** 48–61
[6] Melnik E V and Klimenko A B 2016 *Proc. 10th Int. Conf. Appl. Inf. Commun. Technol.* (Baku) 492–6
[7] Greenfield D 2016 Fog Computing vs. Edge Computing: What's the Difference? *Autom. World* (URL: https://www.automationworld.com/fog-computing-vs-edge-computing-whats-difference)
[8] Kotov V, Melnik E, Sherbinin I and Korovin I 2009 Raspredelennaya informatsionno-upravlyauschaya sistema na osnove intellectualnih datchikov 2
[9] Melnik E, Klimenko A and Korobkin V 2015 *10th Int. Conf. Appl. Inf. Commun. Technol.*(Rostov-on-Don) (New York:IEEE) 84–95
[10] Melnik E and Klimenko A 2017 *Proc. 11th Int. Conf. Appl. Inf. Commun. Technol.* (Moskow) (New York:IEEE) 355-9
[11] Melnik E, Klimenko A, Schaefer G and Korovin I 2017 *6th Int. Conf. ICIEV*
[12] Melnik E, Klimenko A and Klimenko V 2015 *Proc. 9th Int. Conf. Appl. Inf. Commun. Technol.*(Rostov-on-Don) (New York:IEEE) 277-80