Applications for Monitoring and Visualizing Events from the Cloud or Fog Environment

Nataly Zhukova
Saint-Petersburg Federal Research Centre
of the Russian Academy of Sciences
St. Petersburg, Russia
Nazhukova@mail.ru

Alexey Subbotin
St. Petersburg State
Electrotechnical University "LETI"
St. Petersburg, Russia
Alesu1543@gmail.com

***Abstract*—This article discusses applications for monitoring and visualizing events from a cloud or fog environment. Applications where images are processed by machine learning algorithms for intelligent video surveillance systems are considered. A review of analogues of programs that implement the concept of the Internet of Things has been made. Examples of the real use of ready-made systems in various fields are considered. The problems of using systems with different architectures are discussed. A proposed solution is in the modification of the generalized architecture of the applications for monitoring and visualizing events. It allows improve events monitoring based on the ability to run applications on devices with different technical characteristics and under different operating systems. The solution is shown in the example of ensuring the safety of escalator equipment. Three applications have been developed for a smartphone, a desktop and a website for the visualization of events from a cloud or fog environment. The developed applications were tested on a large amount of data (video information). Specific examples are given.**

I. Introduction

At the moment, intelligent video surveillance systems are widely used in various fields (https://felenasoft.com/xeoma/, https://www.ispyconnect.com/, https://www.sighthound.com/): traffic monitoring, ensuring security in kindergarten, school, university, subway, clinic and/or hospital. The video surveillance operator is now assigned a controlling function: he or she must check the reliability of the event and confirm or decline the assumption that was made by artificial intelligence based on machine learning [1]. The interaction system of the intelligent video surveillance operator is carried out through applications for monitoring and visualizing events from the cloud or fog environment, both through a website with a personal account and through a separate application for a desktop or tablet [2].

The processing of information received through CCTV (Closed Circuit TeleVision) cameras is usually executed at different levels, which is caused by the low computing power of embedded technological computers and CCTV cameras themselves [3]. This may be a fog computing environment in a neighboring room of a small organization or, if it is impossible to get by with a fog computing environment at the system design stage, a cloud on a server in a data center in another country: Holland, Estonia, Germany, Sweden, etc., which can be very far from the video surveillance system, but in this case, fine-tuning routing is required to reduce communication delays (flickers).

Such data processing systems have a three- and four-tier architecture, where the first tier is a tool for monitoring and visualizing events (PC, smartphone, built-in technological computer with a screen and support of a touchpad, joystick, steering wheel, buttons and other possible controls), the second tier is a high-performance cloud for image processing, and the third is an information warehouse with a data lake. Sometimes there is a fourth tier or instead of the second - a fog computing environment [4], [5].

A fog environment differs from a cloudy environment in several ways:

- proximity to devices with programs for visualization and information collection (sometimes video cameras are connected directly to the fog server via coaxial cables or twisted pairs via expansion cards and Wi-Fi);
- relatively low performance compared to cloud servers, but higher performance than PCs;
- the possibility of secure data storage, confirmed by security certificates from the manufacturer of servers for fog environments, which is sometimes necessary when working with medical data (fluorographic images, etc.), where unsupervised use is prohibited in many countries without permission, as well as the safety of state secrets, commercial secrets and copyright;
- availability of customized software (drivers, limited functionality for the developer - API and development tools, diagnostic utilities, preinstalled programs and libraries for machine learning, customized Linux operating system, etc.);
- the possibility of cluster organization of servers for fog computing, where each server is a separate unit, and the entire computing system can be scaled.

The problem is the incompatibility of the technical characteristics of computers and devices. A new approach is needed when applications work with a fog environment from different devices using different technologies. At the same time, fog computing should be used to accelerate image processing based on machine learning in order to improve the efficiency of intelligent video surveillance.

ISSN 2305-7254

II. Background

Applications for monitoring and visualizing events [6] can be divided into several types depending on the technologies used:

• website (usually PHP, JavaScript, JQuery, React, Node.js, Vue.js, Bootstrap, Laravel, Symfony, WordPress, Moodle, etc.);

• application for PC under Windows, Linux, MacOS (in development environments: NetBeans, Qt, Visual Studio, Code Gear; in languages: Java, C++, C#);

• smartphone application for Android, iOS, Symbian, etc.

Applications for smartphones (having extensions: *.apk, *.ipa, *.sis, *.sisx, *.jad, *.jar) can be divided into several main classes:

• wrapper programs for websites, which in fact are not separate applications but use a separate component on the "browser" form, where a specially prepared page opens (either in the phone's memory or on the server using the protocols: HTTP/HTTPS);

• programs that run and work directly on a smartphone and can work offline;

• programs for interacting with remote servers (clouds, fog environments), where part of the program logic is executed on a smartphone, and the second part on a server for data processing.

An application on a smartphone [7] for monitoring and visualizing events assumes that the program on the smartphone is constantly in memory and periodically (1 time in N seconds, usually 10 seconds) accesses the server, where the flag state is set in the cloud (usually it is file) about the occurrence of a new event. To visualize an event on a smartphone, a status bar is provided (at the top of the screen), and in active mode, the monitoring events are visualized directly on the main screen.

Recently, smartphone apps have become more popular than binaries (*.elf, *.exe, *.app) for desktop Linux, Windows, and MacOS. The principle of operation of such applications is based on a permanent connection to a remote server via the Telnet protocol with sending commands from the server to the client program to receive information about the event. Transfer of archives with information, as a rule, is carried out with TLS/SSL encryption protocols [8].

The website [9] works in a similar way, where the visualization of events from surveillance cameras is updated in a frame after entering the personal account. Since constantly updating the page takes up much browser memory, the UDP/RTP protocol is used without error control. Basic information is transmitted via HTTP/HTTPS protocols.

A generalized scheme for visualizing and monitoring events (Fig. 1) can be represented as three main blocks: technological computers in communication nodes (1), desktop programs for PCs (2) and Internet of Things devices (3), where smart watches are located, touch pads, and smartphones (lower right corner in Figure 1). In the center are fog computing environments, which are not always present in the architecture and greatly reduce the delays when accessing the cloud. In the cloud server, images are directly processed using machine learning (ML) algorithms, and information is stored in storage, which is most often represented as a data lake [10]. This form of storage is recommended by MEC vendors and is often used in off-the-shelf solutions of various intelligent video surveillance systems.

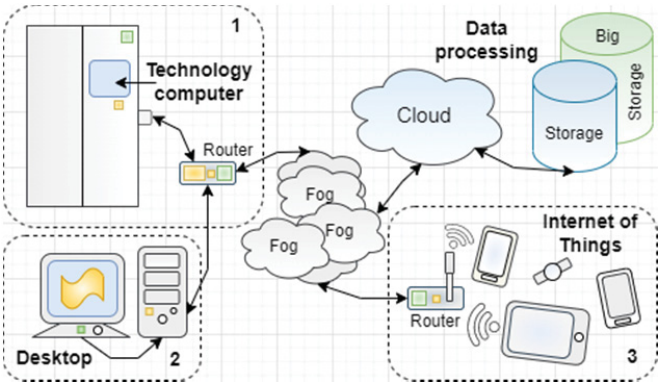

Fig. 1. Generalized scheme for the visualization and monitoring of events

The event visualization program is an integral component of the Mobile Edge Computing (MEC) concept, which is widely represented by various vendors, such as Microsoft Azure MEC, Intel Smart Edge, Accenture, Cognizant, HCL, Tech Mahindra and many others [11]. Now, the MEC concept is used almost everywhere: from fast food chains (McDonalds, KFC, Burger King, Dodo Pizza, etc.), to oil rigs and medical operations [12]. In some cases, solution providers provide a separate platform for organizing edge computing; in other cases, it is a hardware-software implementation of the MEC concept at the device level. For example, Huawei (https://www.huawei.com/) introduced its processors (GRU, NP and AI chipsets) with support for edge computing. Applied adaptation for the interaction of 5G devices in B2C and B2B format for Huawei Packet Core Network programs is provided, reducing the load on computing directly on the device. Another ASUS IoT company (https://iot.asus.com/) presents ready-made device-level solutions (technological computers, industrial motherboards, routers, small-scale storage, etc., devices) with ready-made solutions in the field of artificial intelligence, machine learning, network security and Android FOTA support for millions of devices (smart badges, video cameras, industrial robots, etc.).

A platform based on the MEC concept provides only an approach and basic tools for working with edge computing, and the creation of an intelligent video surveillance system and a program for visualizing and monitoring events is reserved to the developer.

III. Description of the Problem

The main problems [13], [14], [15] of applications for visualization and monitoring of events from an intelligent observation system that is located in a cloud and/or fog are as follows:

- infinitely large video stream traffic from different surveillance cameras for one device [16];
- reliability of events [17] during frame-by-frame demonstration of an image with a certain time interval (from 10 seconds or more);

- absence of a fog environment and, as a result, strong communication delays (flickers);
- security of the system if the images are classified (commercial, state, military, medical, etc.) and protected by copyright [18];
- availability of only one visualization device: API for development environment for desktop (Windows 11, Linux KDE, MacOS Leopard), smartphone (Android or iOS), technological computer (Advantech, Siemens);
- synchronization of information about one object from different surveillance cameras into one event (robbery, fire, terrorist threat, etc.);
- synchronization restoration of the retrospective and chronology of events from one object received from different sources and places of observation (checkpoint, corridor, workshop, locker room, dining room, pantry, crime, fixation of drunken robbery, etc.).

The obsolescence of software and the lack of image processing using machine learning [19] in monitoring and control systems for various metropolitan infrastructure facilities (railroads, schools, supermarkets, subways, clinics, etc.) motivate the modernization of existing security systems to improve their effectiveness. A good applied task for new programs for the visualization and monitoring of events can be a monitoring system for escalator equipment in the subway, where the length of escalators can reach up to 140 meters. Cases of accidents and catastrophes that happen in different subways of the world are not uncommon. The reason for this was a malfunction of the brakes, electrics, engines. At the moment, a frequency converter (FC) is used without indication of the states of the complex of escalator systems. The incoming high-frequency current from the municipal power grid is converted by the inverter into the permissible current for the operation of motors (50 Hz, 380 and 220 W). A modern status indication and a visualization and monitoring program with event diagnostics are required for the Advantech embedded technological computer, which supports control and configuration via the touch screen. On the panel computer, it is possible to diagnose engine malfunctions using machine learning (ML), recognize illegal entry into the service premises of the escalator and attempt to open control cabinets. It is necessary to visualize diagnostic information from the centralized power grid and urban substations.

At the moment, monitoring the status of the escalator equipment is insufficient: there is an indication panel where information is received from sensors, there is video surveillance from video cameras on the TV screen received via coaxial cables, but there is no image recognition, which would greatly simplify the work and increase the efficiency of the security service at the subway. It is necessary to modernize not only the FC cabinet but also the workplace of the escalator equipment dispatcher. It should be a modern visualization application (a website with the ability to view real-time statistics). There is a need in a desktop application with low requirements on computing resources. Since the website cannot be run on the Advantech TPC-61T-E3AE technology computer, an application (GUI) for the KDE or Gnome desktop, designed in GTK and Qt, is needed.

A truly modern way to monitor and visualize events can be implemented as a smartphone application, which was developed in Java in the IntelliJ IDEA development environment for mobile programs from JetBrains (https://www.jetbrains.com/idea/). For development, the Ultimate 2021.3 version was chosen since it supports not only the Android mobile platform, for which the APK application is being developed but also JavaScript, SQL, Ruby, PHP, which are necessary for storing statistics in the smartphone's memory, the cloud, and interacting with the fog environment [20], [21], [22] and the logic of displaying block elements on forms in the application. Platforms were chosen as a cloud computing and data storage system: mClouds (https://mclouds.ru/) and Selectel (https://selectel.ru/) with the possibility to connect via remote desktop to a VMware virtual machine. The chosen configuration was 8vCPU 3.1-3.9 GHz, 32 GB RAM, 110 GB SSD with Ubuntu VPS/VDS operating system on mClouds platform and 50 TB data lake in hybrid cloud on Selectel platform with migration capability and high fault tolerance for machine learning.

Applications from different technology stacks are needed to monitor and visualize events that solve the following tasks:

- indication of events in a single complex (one system) with an artificial intelligence (AI) system based on neural networks;
- pattern recognition in the video information stream based on machine learning;
- processing images with the definition of the type of security situations;
- indication of escalator equipment (motors, pumps, electrics, alarms, video cameras, etc.);
- provision of centralized information from substations.

What is needed is a comprehensive assessment of situations based on information from deterministic sources, generalized in a single program.

IV. Solution method

The solution of the considered problem is the development of a new system for visualizing and monitoring events in the escalator's electrical network, consisting of three programs: a website, an application for Linux KDE (*.elf) and a smartphone application (*.apk) for the Linux operating system Android 10/11 (https://www.android.com/) because this OS is more popular than iOS.

A constructive modernization of the inverter is needed, adapted to global trends. There is a need in a screen for diagnosing and monitoring the status of the electrical system of the escalator. The solution at the design stage of a new inverter unit is to add an Advantech (https://www.advantech.com/) TPC-61T-E3AE embedded process computer in an impact-resistant housing (ABS) and IP65 screen protection to the wall of the inverter housing. The TPC-61T-E3AE or PPC-3120-RE9A series touch panel is suitable for demonstrating production processes and monitoring and visualizing events from a cloud or fog environment [23], [24], [25]. However, a better decision is to purchase a PPC-3211W series process computer since the technical characteristics (memory, processor, interfaces, screen characteristics, etc.) are significantly higher than those in the previous models. The KDE Linux desktop application should be installed on a 512 MB MicroSD card and run within 2-5 minutes, which includes

Linux boot time with hardware test function, X-Window (/etc/X11/fly-dm/Xsession) and shell startup KDE with autologin (/fly-dmrc), where from autoload (/etc/rc.local) the application for monitoring and visualizing events is launched.

The outdated PC program needs to be replaced with a modern website that supports HTML5 and CSS3.2. Unlike a Windows desktop program, a website can be opened from any computer on the network and even a tablet and/or smartphone, regardless of the operating system (Android, iOS, Symbian, etc.). In a fog or cloud environment, an Apache web server with a PHP preprocessor on FreeBSD 13.0 (https://www.freebsd.org/) should be installed, which is best for a web server and configured over HTTP via Webmin 1.941. In addition, FreeBSD is installed perfectly on a VMware virtual machine in the cloud without additional drivers. Creating dynamic pages from the "htdocs" directory is done using Perl 5.34.0 (https://www.perl.org/get.html) and in some cases PHP 8.0.13 (https://www.php.net/), AJAX is used for loading content (in the formats: JSON, XML), and JavaScript and the jQuery 3.3.6 library (https://jquery.com/) are used for supporting menus and logic on the website.

The most popular development environment for mobile devices is Android Studio 4.1.3 (https://developer.android.com/studio) from Google. The development of programs for Android and iOS is carried out in the cross-platform language Dart with the ability to build applications for different operating systems. However, the application requires a set of API functions to interact with the SQL database, so the IntelliJ IDEA Community Edition development tool was chosen with the ability to upgrade to Ultimate 2021.2.3. Networking with the cloud or fog environment is organized through the Maven web server and the Swagger framework for exchanging information about events in XML and JSON, the profile of which is uploaded to the SwaggerHub service (https://swagger.io/), which provides high-speed interaction and collaborative development. The SwaggerHub service supports OpenAPI Development (except native: SwaggerAPI and SwaggerUI) to simplify the development of APK network applications for mobile devices (Samsung, Nokia, Xiaomi, HUAWEI, DEXP, Itel, Tecno, TCL, BQ, Blackview, INOI, Infinix, Vivo, realme, Oukitel, OPPO, Motorola, Doogee, Honor, etc.).

After brainstorming and discussion with subject matter experts and management, nine expected performance indicators (Table I, first column) and expected outcomes (second column) were defined.

TABLE I. REQUIRED INDICATORS

	Indicator	Expected Result
1.	Reduction of escalator failures	20%
2.	Increase of the accuracy of event detection	25%
3.	Decrease of requests for equipment repair	35%
4.	Efficiency of decision-making	30%
5.	Speed of interaction	10%
6.	Reduction the priority of ancillary services	15%
7.	Satisfaction with the system	40%
8.	Accuracy in making managerial decisions	15%
9.	Increase of the efficiency of the department	10%

The architecture of the new event visualization and monitoring system consists of three types of programs, which are central to the intelligent video surveillance system. When the application receives a signal from a fog environment or a cloud (a flag file that signals an event found based on complex information from various things: video cameras, sensors, centralization and power supply management systems, disconnection of supply feeders, substations, malfunctions of engines, pumps, etc.), then a certain decision is made: receiving a signal state in the application of the operator on a smartphone [26], a technological computer in the FC or on the PC screen. The scheduling is used to determine the least loaded route to the cloud or fog environment, determine the busyness of the node and reduce delays (flickers) when choosing an appropriate route to the image processing server. Thus, a highly deterministic system with a high degree of reliability is obtained [27], [28], [29].

According to the TOGAF architecture description standard, the first level of Business assumes a strategy to reduce injuries and accidents in the metro area and to increase the accuracy of identifying situations, making decisions quickly and responding quickly. At the Data level, a description of the interaction in the system at the data level is assumed. In this system, information is exchanged according to the principle of signals and scheduling theory with information processing on the server and storage in the cloud. At the Application level, an intelligent video surveillance system interacts with employees and management through three types of applications (mobile, desktop, website), where the current situation is displayed. At the Technology level, applications are developed in Java, Qt, PHP, etc., using libraries for network interaction.

The ISO 15288 standard is supported, the target system is the application, the system in the production environment is the operating system (Android, Ubuntu Linux, FreeBSD), and the provisioning system is the cloud and/or fog environment.

According to the architecture description standard, DoDAF 2.0.2, the operational view is information based on video, sensors, etc., for the operator of the intelligent video surveillance system; the system view is the software of the computer where the application is running; the technical presentation consists of three components: a smartphone, a technological computer and a PC website.

V. APPLICATION DESIGN

The proposed solution is implemented in the form of applications with the ability to measure the defined five indicators (Table I) and compare the expected results with the actual results. A schematic diagram of the software interface has been developed (Fig. 2), which contains menus, settings, statistics and frames with video streams (in the center of the application form) to view events that are generated centrally on the server from various sources: power supply centralization system, engine operation, pumps, sensors and video cameras.

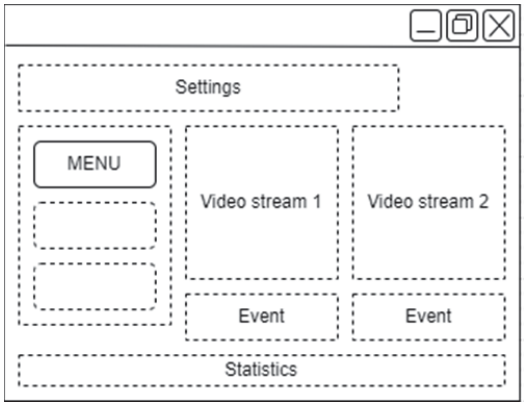

Fig. 2. Software interface diagram

The programs developed by the authors of the article for monitoring and visualization interact with other modules in the cloud or fog environment [30], [31], [32], which also interact with each other using the XML format (RSS 2.0):

1) software control and decoding module (processing information from sensors of the escalator and the centralized power supply system, decoding the video stream into a series of frames, network transmission of information using secure SSL and TLS protocols and sending results to the cloud);

2) a software module for processing information in the cloud (receiving connections via an encrypted VPN channel and establishing connections via SSL and TLS protocols with a list of trusted IP addresses, unpacking an archive with information, processing a series of frames, transferring the results to the fog environment, and then to the devices for event visualization).

A series of processed images (in the format: Portable Network Graphics, *.PNG) obtained from various sources (cameras) with additional information are packed into packages (archives) by the 7-Zip 21.04 program (https://www.7-zip.org/) with passing parameters via the console to script files in BASH 5.1 (*.SH) and are unpacked with the Gzip 1.8 program (http://www.gnu.org /software/gzip/), since it is adapted for Ubuntu Linux and FreeBSD; therefore, it works faster than 7-Zip and other archivers. Packets are transmitted using the http4you 0.37 utility from the rss2fido 0.52 software package (http://rss2fido.sf.net/) for downloading and modifying XML files after receiving a status signal on the server about the event found for the visualization program on the device of the intelligent video surveillance operator.

VI. Case Study

Three applications have been developed for monitoring and visualizing events: a website, a Linux desktop program KDE, and an Android mobile application.

The website is running FreeBSD 13.0 under a Nginx 1.25.0 web server (https://nginx.org/) but can be ported to any other system that supports a PHP parser. The website is hosted on the local machine (http://localhost) and opened in the Google Chrome browser. Such a site can be opened on any device (Android, iOS, Symbian) and under any operating system (OpenBSD, CentOS, Astra Linux, Ubuntu Studio, QNX, macOS, etc.). In the center (Fig. 3) there are frames in which PNG images are updated every 3 seconds so as not to heavily load traffic and not freeze the browser page. At the bottom of each frame is an indication panel. In this case, all is well (Event: Good). In case of a dangerous situation, there will be messages: Alarm, Alert, Trouble, Worry, Error - depending on the degree of danger determined by artificial intelligence.

On the right side of the website page there is a menu (Fig. 3), switching on which allows watch the stream (tab: View) with visualization of events for the video surveillance operator. Next, for the operator select views and angles (tab: Choose), configure the parameters of video cameras (tab: Cameras), sensors - their parameters and connection (tab: Sensors), test equipment and identify faults (tab: Test). Next, configure the network (tab: Net) and obtain the support of a technical specialist (tab: Support). In the upper right corner are shimmering cubes, signaling the state of the connection with a cloud or fog environment. The design of the site follows the interface scheme (Fig. 2) and has the HTML5 format based on the generalized DOM structure of the HTML4 sites.

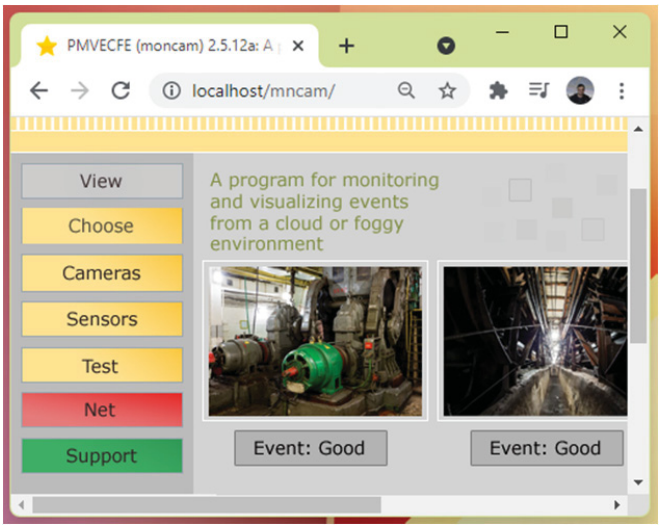

Fig. 3. Appearance of the website for monitoring and visualization of events

The Linux application (MnCam2.4.7c.elf) for the KDE desktop is required for technology computers (Advantech, Siemens, etc.) where there are severe limitations on computing resources. The source files of the project were compiled under Windows 11. In the center of the main form of the application (Fig. 4), there are two video streams (Stream 1 and 2) from surveillance cameras obtained from the cloud or fog, with the ability to switch up to 5 objects (selection on the right) and up to 9 threads for one object in one instance of using the Next button. The menu is made in the traditional way for desktop applications in the form of a drop-down list. Additional windows with settings appear from the Window menu item, which repeats the functionality of the pages on the website (Choose, Cameras, Sensors, Test, Net, Support).

The diagnostic messages correspond to the classification as on the website (at the bottom of the images in the Event field). At the moment, everything is fine (the message Event: Good is issued). At the bottom of the main form, there is a statistics field with the ability to search (field: Put search), clear log files (button: Clear), open in a text editor (button: Open) and on a web page (by button: Web) in a convenient form of a table with filtering and advanced search capabilities. The statistics web

page is located on the same server as the event monitoring and visualization web site but differs by only one directory on the server: http://localhost/mncam/stat/ and runs using the same PHP preprocessor.

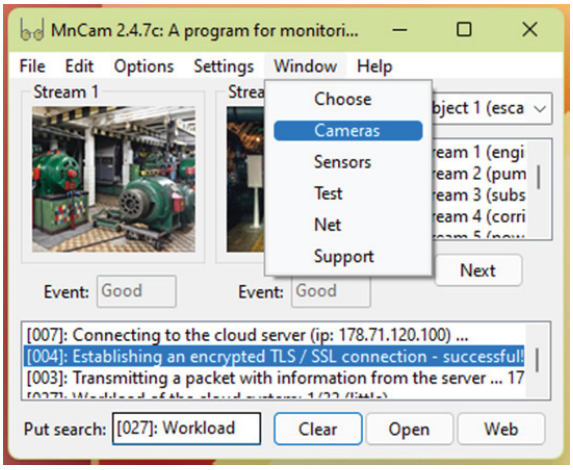

Fig. 4. Desktop App Appearance

A program has also been developed for a smartphone (Fig. 5) in APK format (monCam1.4.7b.apk), which is a standalone application, not a shell with a "browser" component and runs directly on the device, thanks to the IntelliJ IDEA Ultimate development tool, which contains independent components and does not download an HTML site, which, in fact, is not a standalone program but a demonstration of the internal function of the built-in Android browser. After removing the browser from the smartphone's memory, the shell program will not work on its own.

As presented in Fig. 5 Smartphone Application is a separate running application under the Android OS. In the center of the screen is an image of the engine compartment of the escalator in the subway. People were found on the received frame from the cloud, and this event is defined as STRANGERS. The operator of the escalator safety system is prompted to make his choice using the buttons on the display: acknowledge the event (ACCEPT), deny (ERROR) or pay attention (ATTENTION). In the lower right corner of the application, the time of a certain situation (strangers in the office) is fixed. In the left corner, there is a back button that allows you to go to the application settings. This approach is not a limitation of the functionality but, on the contrary, does not distract the security operator from unnecessary settings and indicators (testing the network, equipment components on the server, video cameras, sensors, centralization systems, information from substations, etc.).

The performance of the proposed solution has been estimated in the following conditions:

- 14 streams from Panasonic Full HD HC-V730 cameras at 24.0 pixels;
- Network connection 100 Mbps;
- HPE Apollo 6500 Fog Computing Environment (https://www.hpe.com/);
- Splitting the video stream into a series of 10 frames per second;
- 1920x1080 resolution every frame and 350 dpi depth with full color reproduction and transparency (alpha channel transmission);
- Fog should have HPE Ezmeral Runtime and ML Ops 5.3 from HP installed;
- Samsung Galaxy J3 (2016) smartphone with 1.5GHz ARM Cortex-A7 processor, ARM Mali-400 GPU, 1.5GB RAM and 32GB internal storage.

Fig. 5. Smartphone App Appearance

Nine performance indicators need to be measured. The conditions for repeating the case are the development of three visualization programs, setting up the receipt of batch information about events (a series of processed frames with additional information in XML) from the cloud or fog environment. The first measure of effectiveness is the reduction in the number of escalator failures (Table II). After estimating the results regarding failures in the operation of a group of escalators in a separate section, it was found that the number of failures decreased by 28.98%, although no more than 20% was expected, which is a good result.

TABLE II. FAILURES IN THE WORK OF ESCALATORS

Period	Before	Now	Less
1 Jan. — 31 Mar. 2019	23	16	30.44
1 Oct. — 31 Dec. 2019	19	14	26.32
1 Jul. — 30 Sep. 2020	15	9	40
1 Oct. — 31 Dec. 2020	21	13	38.1
1 Jan. — 31 Mar. 2021	27	17	37.04
1 Apr.— 30 Jun. 2021	18	15	16.67
1 Jul. — 30 Sep. 2021	14	12	14.29
Average (%):			**28.98**

The second measure of effectiveness is the improvement of the event detection accuracy (Table III). After estimating the results regarding the accuracy of identifying events related to the failure of the escalator equipment, it was found that the accuracy was improved by 27.1%, although no more than 25% was expected, which is a good result. Events have become more precise.

TABLE III. EVENT DETECTION ACCURACY

Event	Earlier	Now	More
Fire	0.81	0.94	13%
Lighting off	0.87	0.98	11%
Robbery	0.62	0.87	25%
Illegal entry	0.51	0.96	45%
Fur. breakage or destruction	0.64	0.91	27%
Terrorist threat	0.57	0.89	32%
Flooding	0.73	0.9	17%
Engine destruction	0.68	0.87	19%
Escalator damage	0.52	0.93	41%
Substation accident	0.6	0.95	35%
Power failure	0.57	0.9	33%
By average:			**27.1%**

The third performance indicator is the decrease in the number of requests for equipment repair (Table IV). After estimating the results for the third quarter of 2021 regarding escalator equipment repair requests in a separate section, it was found that the number of requests has decreased by 37.2%, although no more than 35% was expected, which is a good result. There were fewer requests for equipment repair, which means that specialists began to manage on their own for repairs.

TABLE IV. APPLICATIONS FOR EQUIPMENT REPAIR

Title	Previously	Now	More
Engine	4	2	50%
Switch	21	12	42.86%
Cable	5	3	40%
Defense	16	9	43.75%
Grounding	5	4	20%
Bearings	3	2	33.34%
Miscellaneous	39	27	30.77%
By average:			**37.2%**

The fourth indicator of efficiency is the efficiency of decision-making on the repair of escalator equipment (Table V). After estimating the results regarding the efficiency of repairing equipment, which is an integral part of the escalator, it was found that the time was reduced by 31.07%, although no more than 30% was expected, which is a good result. The speed of repair and commissioning of the escalator has increased, which increases the satisfaction and quality of the escalator system.

TABLE V. EFFICIENCY OF REPAIR

Title	Previously	Now	More
Engine	8-10 hours	6-9 hours	17.4%
Switch	15-20 min.	12-17 min.	23.82%
Cable	4-7 hours	2-5 hours	48.2%
Defense	30-50 min.	20-35 min.	39.51%
Grounding	2-3 hours	1-2 hours	49.73%
Bearings	5-8 hours	4-7 hours	12.1%
Miscellaneous	20-60 min.	15-45 min.	26.7%
By average:			**31.07%**

The fifth indicator of efficiency is the speed of interaction between subway support services in orser to repair the escalator equipment (Table VI). After estimating the results regarding the time spent on interaction in the elimination of failures in the operation of escalator equipment in a separate section, it was found that the time increased by 13.68%, although no more than 10% was expected, which is a good result. Reduced interaction time with firefighters, police and many support services.

TABLE VI. SPEED OF INTERACTION

Service	Before	Now	More
Police	10-15 min.	7-12 min.	12.7%
Ambulance	5-7 min.	2-4 min.	23.18%
Fire department	12-17 min.	10-15 min.	8.29%
Tap water	20-30 min.	15-25 min.	6.851%
Electric	15-25 min.	12-20 min.	7.31%
Mechanical	40-60 min.	30-50 min.	5.94%
Security	2-5 min.	1-3 min.	31.46%
By average:			**13.68%**

The sixth performance indicator is the deprioritization of support services, self-repair (Table VII). After estimating the results regarding the priority of support services in the repair of escalator equipment in a separate section, it was found that the indicator decreased by 16.51%, although no more than 15% was expected, which is a good result. The priority of auxiliary services has decreased due to the new system.

TABLE VII. ANCILLARY SERVICES PRIORITY

Service	Before	Now	Less
Police	79.1%	67.82%	11.28%
Ambulance	68.31%	52.7%	15.61%
Fire department	52.6%	37.2%	15.4%
Tap water	67.9%	51.42%	16.48%
Electric	72.41%	53.4%	19%
Mechanical	61.2%	47.23%	13.97%
Security	58.4%	34.61%	23.8%
By average:			**16.51%**

The seventh performance indicator is the overall satisfaction with the system and the ease of use of applications. Satisfaction with the system (Table VIII) was higher than expected and amounted to 63.2% versus 40% previously estimated, relative to negative response options. A standard psychological questionnaire was used, consisting of 100 open-ended questions to assess attitudes towards the application. The passage was organized through Google Forms online in free time (https://forms.google.com/).

TABLE VIII. SATISFACTION WITH THE SYSTEM

Position	Agree	Against	False	Neutral	Better
Duty	43	21	4	32	51.2%
Dispatcher	52	17	6	25	67.31%
Electrician	37	14	2	47	62.2%
Hydraulic	29	21	7	43	27.6%
Operator	31	8	13	48	74.2%
Security guard	17	3	29	51	82.36%
Passenger	62	14	5	19	77.5%
By average:					**63.2%**

The eighth performance indicator is the accuracy of managerial decision-making (Table IX). The accuracy of

managerial decision-making can only be assessed by the manager. A standard Google calendar (https://calendar.google.com/) was used to record management decisions and for their further analysis. Despite the fact that it was planned to increase the accuracy of managerial decision-making by 15%, it turned out to be higher and amounted to 18.62%, which is a good result. Management made fewer mistakes and gained confidence in their decisions.

TABLE IX. ACCURACY IN MAKING MANAGEMENT DECISIONS

Solution	1 week	2 weeks	1 month	3 months	More precisely
Dismissal	0.015	0.023	0.034	0.136	11.1%
Retraining	0.13	0.168	0.221	0.82	15.9%
Job transfer	0.041	0.055	0.075	0.31	13.3%
Rebuke	0.087	0.124	0.174	0.692	12.6%
Hiring	0.152	0.206	0.28	0.87	17.5%
Sick leave	0.28	0.37	0.51	0.861	32.6%
Time off	0.185	0.26	0.37	0.795	23.3%
Vacation	0.236	0.321	0.437	0.84	28.1%
Prize	0.109	0.147	0.2	0.805	13.6%
Meetings	0.292	0.39	0.542	0.856	34.2%
Innovation	0.17	0.251	0.36	0.845	20.2%
Software impl.	0.1	0.138	0.19	0.764	13.1%
Testing	0.104	0.14	0.194	0.84	12.4%
Approbation	0.072	0.1	0.142	0.568	12.7%
By average:					**18.62%**

The last indicator is the performance of the entire department (Table X). After estimating the results, the effectiveness of the entire department (team of repair) in servicing escalator equipment increased by 11.85%, although no more than 10% was expected, which is a good result. The efficiency of the department and individual specialists has increased.

TABLE X. DEPARTMENT EFFICIENCY

Position	Previously	Now	More
Boss	89.7%	96.82%	7.12%
Electricians (3 people)	76.31%	91.37%	15.06%
Programmer	85.4%	94.7%	9.3%
Operator	94.72%	98.35%	3.63%
Mechanic	62.59%	86.72%	24.13%
By average:			**11.85%**

As a result of all the experiments, a summary table was constructed (Table XI) with the obtained and expected results for all 9 items. To reproduce experiments, it is necessary to develop programs for processing statistics or open logs in Microsoft Excel 2019, apply a filter and functions: average, sum, rounding, percentage, and fill in the resulting tables. Computer with specifications: Intel Core i3-10105F, 4x3.7 GHz, 16 GB DDR4, GeForce GTX 1050 Ti, 512 GB SSD with Windows 10/11 preinstalled. A permanent Internet connection with access to Microsoft Azure (https://azure.microsoft.com/) is required.

Over time, in retrospect, the indicators were somewhat higher than expected by experts, but this is normal in the first months of the system's operation. In general, good, positive results were obtained.

TABLE XI. RECEIVED INDICATORS

	Indicator	Expected Result	Received Result
1.	Reduction of escalator failures	20%	28.98%
2.	Increase of the accuracy of event detection	25%	27.1%
3.	Decrease of requests for equipment repair	35%	37.2%
4.	Efficiency of decision-making	30%	31.07%
5.	Speed of interaction	10%	13.68%
6.	Reduction the priority of ancillary services	15%	16.51%
7.	Satisfaction with the system	40%	63.2%
8.	Accuracy in making managerial decisions	15%	18.62%
9.	Increase of the efficiency of the department	10%	11.85%

VII. CONCLUSION

The article discusses applications for monitoring and visualizing events from a cloud or fog environment that allows increase the efficiency of subway services in several indicators. Results were achieved in 9 indicators: reduction of escalator breakdowns by 29%, increase in the accuracy of event detection by 27%, etc.

The authors of the article developed three applications for monitoring and visualizing events: for a smartphone, for a desktop and a website - information about which was obtained from a cloud or fog environment. The stages of the work of programs are considered, and separate details are disclosed. Specific examples of the work of the program to ensure the safety of escalator equipment are given. The novelty of the proposed solution lies in the possibility to work with a fog environment from different applications, regardless of the technical characteristics of the computer and operating systems, and artificial intelligence methods based on machine learning are used. The research materials are images from security cameras and additional information from escalator sensors, power supply centralization systems and substations.

The task of visualizing and monitoring events from a cloud or fog environment is also relevant for many other subject areas. Therefore, for example, in the process of visualizing events from the service premises of escalators, it may be necessary to recognize events on the tape, platform and track area. The developed programs can also be used not only in the subway but also in airports, shopping centers and other public places. The research was supported by the state budget (project No. FFZF-2022-0006).

ACKNOWLEDGMENT

The author expresses his gratitude to the State Unitary Enterprise "Petersburg Subway" (http://www.metro.spb.ru/), FRC RAS (http://www.spiiras.nw.ru/) for scientific support and the opportunity to create the article.

REFERENCES

[1] Zeyu Jiao, Huan Lei, Hengshan Zong, Yingjie Cai, Zhenyu Zhong. Potential Escalator-related Injury Identication and Prevention Based on Multi-Module Integrated System for Public Health. Machine vision and applications. March 2021, pp. 1-17

[2] Yannuzzi M., Milito R., Serral-Graci R., Montero D., Nemirovsky M.: Key ingredients in an IoT recipe: fog computing, cloud computing, and more fog computing. In: 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), December 2014, pp. 325–329 (2014)

[3] Aazam M., Huh E.: Fog computing and smart gateway-based communication for cloud of things. In: 2014 International Conference

on Future Internet of Things and Cloud, August 2014, pp. 464–470 (2014)
[4] Bhatia J., Patel T., Trivedi H., Majmudar V.: Htv dynamic load balancing algorithm for virtual machine instances in cloud. In: 2012 International Symposium on Cloud and Services Computing, pp. 15–20. IEEE (2012)
[5] Saecker M., Markl V.: Big Data Analytics on Modern Hardware Architectures: A Technology Survey, pp. 125–149. Springer, Berlin (2013)
[6] Bonomi F., Milito R., Natarajan P., Zhu J.: Fog Computing: A Platform for Internet of Things and Analytics, pp. 169–186. Springer International Publishing, Cham (2014)
[7] Sarkar S., Chatterjee S., Misra S.: Assessment of the suitability of fog computing in the context of internet of things. IEEE Trans. Cloud Comput. 6(1), 46–59 (2018)
[8] Huang C., Lu R., Choo K.-K.R.: Vehicular fog computing: architecture, use case, and security and forensic challenges. IEEE Commun. Mag. 55(11), 105–111 (2017)
[9] Tanwar S., Tyagi S., Kumar N.: Multimedia Big Data Computing for IoT Applications: Concepts, Paradigms and Solutions, vol. 163. Springer (2019)
[10] Jaykrushna A., Patel P., Trivedi H., Bhatia J.: Linear regression assisted prediction-based load balancer for cloud computing. In: 2018 IEEE Punecon, pp. 1–3. IEEE (2018)
[11] Subbotin, A. Improving the Mobile Edge Computing Architecture Using Fog Computing Environments / A. Subbotin, N. Zhukova, P. Glebovskiy // Conference of Open Innovations Association, FRUCT. – 2021. – No 30. – P. 388-395.
[12] Bhatia J., Dave R., Bhayani H., Tanwar S., Nayyar A.: Sdn-based real-time urban traffic analysis in vanet environment. Comput. Commun. 149, 162–175 (2020)
[13] Bhatia J., Modi Y., Tanwar S., Bhavsar M.: Software defined vehicular networks: a comprehensive review. Int. J. Commun. Syst. 32(12), e4005 (2019)
[14] Dastjerdi A.V., Gupta H., Calheiros R.N., Ghosh S.K., Buyya R.: Fog computing: principles, architectures, and applications. CoRR, abs/1601.02752 (2016)
[15] Liu Y., Fieldsend J.E., Min G.: A framework of fog computing: architecture, challenges, and optimization. IEEE Access 5, 25445–25454 (2017)
[16] Dragi Kimovski, Roland Math: Cloud, Fog or Edge: Where to Compute? IEEE Internet Computing 2101.10417, 1–8 (2021)
[17] Khaled Matrouk, Kholoud Alatoun: Scheduling Algorithms in Fog Computing: A Survey. International Journal of Networked and Distributed Computing 9(1); January (2021), pp. 59–74.
[18] Zeeshan Ali, Shehzad Ashraf Chaudhry, Khalid Mahmood, Sahil Garg, Zhihan Lv, Yousaf Bin Zikria. A clogging resistant secure authentication scheme for fog computing services. Computer Networks. Volume 185, 11 February 2021, 107731 [https://doi.org/10.1016/j.comnet.2020.107731].
[19] Ahmad Raza Hameed, Saif ul Islam, Ishfaq Ahmad, Kashif Munir. Energy- and performance-aware load-balancing in vehicular fog computing. Sustainable Computing: Informatics and Systems. Volume 30, June 2021, 100454 [https://doi.org/10.1016/j.suscom.2020.100454].
[20] Vasileios Karagiannis, Stefan Schulte. Distributed algorithms based on proximity for self-organizing fog computing systems. Pervasive and Mobile Computing. Volume 71, February 2021, 101316 [https://doi.org/10.1016/j.pmcj.2020.101316].
[21] Wanchun Dou, Wenda Tang, Bowen Liu, Xiaolong Xu, Qiang Ni. Blockchain-based Mobility-aware Offloading mechanism for Fog computing services. Computer Communications. Volume 164, 1 December 2020, Pages 261-273 [https://doi.org/10.1016/j.comcom.2020.10.007].
[22] Mandeep Kaur, Rajni Aron. Energy-aware load balancing in fog cloud computing. Materialstoday: Proceedings. 17 December 2020. [https://doi.org/10.1016/j.matpr.2020.11.121].
[23] Raafat O. Aburukba, Taha Landolsi, Dalia Omer. A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices. Journal of Network and Computer Applications. Available online 4 February 2021, 102994 [https://doi.org/10.1016/j.jnca.2021.102994].
[24] Maria Gorlatova, Hazer Inaltekin, Mung Chiang. Characterizing task completion latencies in multi-point multi-quality fog computing systems. Computer Networks. Volume 181, 9 November 2020, 107526 [https://doi.org/10.1016/j.comnet.2020.107526].
[25] Fenghe Wang, Junquan Wang, Wenfeng Yang. Efficient incremental authentication for the updated data in fog computing. Future Generation Computer Systems. Volume 114, January 2021, Pages 130-137 [https://doi.org/10.1016/j.future.2020.07.039].
[26] Changhao Zhang. Design and application of fog computing and Internet of Things service platform for smart city. Future Generation Computer Systems. Volume 112, November 2020, Pages 630-640 [https://doi.org/10.1016/j.future.2020.06.016].
[27] Masoumeh Etemadi, Mostafa Ghobaei-Arani, Ali Shahidinejad. Resource provisioning for IoT services in the fog computing environment: An autonomic approach. Computer Communications. Volume 161, 1 September 2020, Pages 109-131 [https://doi.org/10.1016/j.comcom.2020.07.028].
[28] Sunday Oyinlola Ogundoyin, Ismaila Adeniyi Kamil. A Fuzzy-AHP based prioritization of trust criteria in fog computing services. Applied Soft Computing. Volume 97, Part A, December 2020, 106789 [https://doi.org/10.1016/j.asoc.2020.106789].
[29] Hadi Zahmatkesh, Fadi Al-Turjman. Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview. Sustainable Cities and Society. Volume 59, August 2020, 102139 [https://doi.org/10.1016/j.scs.2020.102139].
[30] Julian Bellendorf, Zoltán Ádám Mann. Classification of optimization problems in fog computing. Future Generation Computer Systems. Volume 107, June 2020, Pages 158-176 [https://doi.org/10.1016/j.future.2020.01.036].
[31] Xincheng Chen, Yuchen Zhou, Long Yang, Lu Lv. Hybrid fog/cloud computing resource allocation: Joint consideration of limited communication resources and user credibility. Computer Communications. Volume 169, 1 March 2021, Pages 48-58 [https://doi.org/10.1016/j.comcom.2021.01.026].
[32] José Santos, Tim Wauters, Bruno Volckaert, Filip De Turck. Towards end-to-end resource provisioning in Fog Computing over Low Power Wide Area Networks. Journal of Network and Computer Applications. Volume 175, 1 February 2021, 102915 [https://doi.org/10.1016/j.jnca.2020.102915].