Intelligent Browser Window System for Visualizing Events Detected from Video in Distributed Fog Computing Environment

Alexey Subbotin[1(✉)], Nataly Zhukova[2], and Elena Stankova[3]

[1] Saint-Petersburg State Electrotechnical University, Professor Popov Street 5, 197376 St. Petersburg, Russia
alesu1543@gmail.com

[2] St. Petersburg Federal Research Center of the Russian Academy of Sciences, 14Th Line V.O. 39, 199178 St. Petersburg, Russia

[3] St. Petersburg State University, 7-9, Universitetskaya nab., 199034 St. Petersburg, Russia

Abstract. The article discusses a distributed container-based fog computing that powers the window system. Distributed fog computing can significantly increase the speed of window system rendering. The article proposes an intelligent window system for a browser for visualization events detected in a fog computing environment. An overview of visualization tools and settings for video servers is presented. Disadvantages of existing solutions are identified. A new system for visualizing events and editing settings in a browser for a video server in fog is created. The novelty lies in the fact that windows with information in the browser are formed by a neural network based on the user's preferences defined using personal information that is collected with user's consent. The window system in the browser is mobile, high-speed and universal and supports the possibility of configuration. The proposed system was tested in subway subject domain. Its usage made it possible to increase the reaction rate of personnel in the subway by 20.3%. The created window system for the browser can be applied in different areas: airports, shopping centers, hospital buildings, cruise liners and other areas of limited space to improve the consistency of services and the speed of staff interaction.

Keywords: Fog computing · Distributed computing · Window system creation · Event visualization · Intelligent subway video surveillance system · Intelligent assistant · Event and pattern recognition

1 Introduction

Today it is important to create a window system in the browser, as the standard window systems have a long history and are overloaded with a huge number of service functions. Popular window systems are not modern and do not support the intelligent video surveillance system at the core, in particular, event visualization and neural networks. Support for a video server in a fog computing environment should be a priority, in the core of the window system in the browser.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15894, pp. 309–324, 2026.
https://doi.org/10.1007/978-3-031-97648-3_21

1.1 History of Window Systems

The first window system appeared in the Multics-MIT operating system in 1964 for the monitor, before that, in 1955, the GM-NAA operating system was created for the printer for the GE 645 computer in 1955. The window system for the monitor running Multics resembled a space divided into parts from four green windows. There was a primitive status line at the top of the screen. The size of its source code in assembly language is only 120 thousand lines and is available for study (https://www.multicians.org/). The second popular UNIX system appeared in 1970, it is distinguished by minimalism and does not have even a primitive window system in the kernel, but only a command line due to the limitations of PDP-11 and VAX computers. The size of the UNIX kernel is only about 10 thousand lines and is available for study (https://github.com/dspinellis/unix-history-repo). The minimalistic UNIX kernel is the basis for popular operating systems: Linux, BSD, Solaris, QNX, MacOS and Windows, starting with version XP SP3, and version 11 supports the Linux kernel through the WSL command with display of Linux applications on a single Windows desktop. The first version of Linux 0.01 (https://github.com/zavg/linux-0.01) appeared in 1991 based on MINIX (https://www.minix3.org/), each system contains about 10 thousand lines of code, which is comparable to the size of the program code of a small website. The author of the MINIX operating system describes in detail the functions of the kernel and working with memory, that any programmer in the C language can create an operating system in six months (the first version of Linux based on MINIX was created in six months) [1]. The main difficulty in creating an operating system is device support (creating drivers). UNIX utilities for working with memory, directory structure, file system are very simple and can be easily recreated in the C language. The X-Window graphical subsystem is designed in such a way that it can work with drivers and kernel functions of any UNIX-compatible operating system (Linux, BSD, etc.). Like Windows 3.11 and Windows 95, the X.org window system based on the X11 protocol is an application to the UNIX-class operating system, just like Windows 95 and DOS 6.22. The X-Window window system provides only basic functions; to interact with programs, it needs to install a desktop environment (for example, GNOME or KDE based on widgets: GTK + or QT) [6–8].

1.2 Distributed Computing

Fog and distributed computing can significantly increase the speed of an application, in particular, it can increase the speed of information processing (video images, cryptocurrency mining, data processing for intelligent video surveillance systems, etc.). Visualization of events in a window system in a browser can also work faster using distributed computing. Computations can be executed both on containers and on virtual machines. In distributed computing the load is distributed dynamically depending on the load of each container or virtual machine. The manager of such computing is considered to be an orchestrator program. A popular program for distributed computing is Docker (https://www.docker.com/), which is a container manager similar to virtual machines (PC), but it interacts with the operating system environment. It is possible to transfer files and directories, devices and much more from the host system directly at the start of the program. The container can be in the cloud, a local computer or a fog server.

The manager can work only with one type of container. For example, Docker work only with DockerImage and sends containers to DockerHub using the Dockerfile utility. An example of creating a container: “docker container create -a subcontwin”.

Many software manufacturers abandon containers in favor of virtual machines due to greater isolation and code security. The speed of a virtual machine is sometimes no different from containers, and sometimes it surpasses when starting a virtual machine from sleep mode, which is not supported by container technologies. A window system in a browser can greatly speed up work using distributed computing technologies due to working on a series of virtual machines managed by an orchestrator. Popular virtual machines are VirtualBox (https://www.virtualbox.org/), VMware (https://www.vmware.com/) and Parallels Desktop (https://www.parallels.com/), which can run under popular operating systems: MacOS Sequoia, Windows 11, Linux Debian 12.

1.3 Modern Browser Interfaces

The majority of existing window systems are programs loaded from the command line or they are automatically loaded on top of the operating system. However, starting with Windows 98, Microsoft developers made deep integration of Explorer with Windows and Explorer became a windowing system [9]. After loading drivers and checking devices, Internet starts Explorer 5 (or MSIE 6.0), which opens a site with a window system. Up to the latest version of Windows 11, a developer can install an application that will patch the Windows kernel, which will allow to minimize the desktop to the taskbar, maximize it to a window on a blue background, and maximize it to full screen, like full-screen mode on the Internet Explorer using F11 button. Theoretically, instead of Internet Explorer one can download Mozilla Firefox, Google Chrome or any other browser with a website that simulates a window system. The Explore window system website can be loaded from the local disk “C:\Windows\System32” (the default, set by Microsoft developers Windows) or a webserver, such as “http://localhost/windows”. It is not difficult to create a site that imitates the window system in a browser, but according to Bill Gates, full duplication of the Windows window interface is prohibited by Microsoft copyright. There are several window systems in the Internet that work instead of the Explore task, which is a running copy of Internet Explorer with a window system. After the Explore task is destroyed, the desktop and all applications completely disappear from the screen. It is possible to restart a website with the Windows window system (taskbar, shortcuts, etc.) by running the “explorer” command in the command line, after which the zero task Internet will start Explorer with desktop and all applications [10–12].

The window system in the browser does not necessarily have to be created using ready-made libraries: Bootstrap, Vue.js, React, jQuery and many others, which are common for creating interfaces: buttons, fields, forms, windows and other graphic elements. It is possible to create a window system website on the modern Chromium 44.x engine based on JavaScript, which forms a dynamic DOM on HTML5 and CSS3.2 with dynamic loading through the element “ < iframe > … < /iframe >”, available in new browsers [13–15].

1.4 Video Analytics and Visualization

Hitachi VSP 5500 video servers have pre-installed software from Microsoft: Windows 7 operating system and a program Windows Media Player (WMP) for viewing video files on the server. The video server has the following characteristics: performance of 21 million IOPS (operations per second) and a throughput of 149 GB/s. The video server is very powerful compared to the client computer.

A very popular video server is the Cisco UCS C480 ML M5 Rack Server, which is also used as a server for machine learning and neural networks. The server developers have implemented the Cisco Unified Computing System (UCS) program, which provides virtual environments at the core level of the Intel Xeon Scalable processor with 28 cores and a motherboard for two such processors. The UCS version server supports up to eight NVIDIA SXM2 V100 32G video cards, which can be combined via NVLink. The C480 series server supports up to 7.5 TB of DDR4 2933 MHz RAM and 24 DIMM slots for shared memory. The Cisco ML M5 modification server supports PCI Express (PCIe) 3.0 slots with 10 Gigabit Ethernet LAN-On-Motherboard (LOM) ports and expansion slots for connecting external devices: video cameras, sensors, etc. intelligent video surveillance system devices. The Cisco server has pre-installed programs for combining computing, virtualization, and statistics (UCS Manager, SingleConnect technology, Integrated Management Controller). The Management Software Development Kits (SDKs) and DevOps software package contains examples for working with neural networks and machine learning. The Cisco server comes with detailed documentation (https://developer.cisco.com/).

Another popular video server HPE DL384 Gen12 (https://hpe-servers.ru/) has a NVIDIA GH200 NVL2 Grace Hopper 144GB HBM3e video card, 960GB LPDDR5X RAM with a capacity of up to 1.2 TB and a throughput of up to 5 TB/s. The server is optimized for machine learning (ML), the creation of language models with intensive use of RAM, requiring a large amount of memory. The HPE server is optimized for building very detailed models and forecasting using ML. It has pre-installed Retrieval Augmented Generation (RAG) programs to improve performance. GPU management on the HPE ProLiant Compute DL384 Gen12 occurs through the HPE ProLiaProLiant Compute DL384 program and the HPE iLO profile. The HPE ProLiant server has expansion ports (EDSFF Gen5 x4 NVME, USB 3.1, iLO Service, Serial, VGA, OCP 3.0) for connecting various devices. The NVIDIA GH200 Grace Hopper graphics chip is also a central processor with artificial intelligence (AI) functions with NVIDIA OVX certification.

Existing systems have a number of limitations that appear when they are used for visualization of events that are the low speed of the application, lack of mobility, configurability and lack of universality (work only under Windows 7). The problem is proposed to be solved by creating an intelligent window system in the browser for visualization of events and video server settings in a fog computing environment.

2 Window Systems Overview

A popular program for viewing events from a video server is Milestone XProtect, which is a suite of video analytics programs. For example, the Smart Client 7.0 program provides a list of objects and assigned video cameras, each video camera can be opened in a

conditional workspace divided into four parts. Perhaps this limitation was created by the developers, since high-resolution 4k video with more than four streams at the same time will greatly slow down the client computer. Video recording and saving of individual frames are available. The alarm manager is opened by password, it is possible to create an HTML site with button navigation and a link map based on data from Smart Client. Milestone program Federated Architecture can work with not one, but several video servers at once. XProtect Enterprise version is capable of recognizing audio events. XProtect Express version works with PTZ surveillance cameras. XProtect Go version can save frames in JPEG/PNG format. All versions of XProtect allow to create a map and HTML site with video fragments in the designer mode. Each video camera has three indicators: event, movement and work. It allows configure the quality of the transmitted video MPEG (I-frame) H.264 and JPEG with a frame rate per second (from 0.2 to 20). A detailed guide is available on the website (https://www.milestone-spb.ru/). Currently, the system is used in the State Unitary Enterprise "Petersburg Subway" (https://metro.spb.ru/) [17, 18].

There are a number of other popular programs for viewing events and configuring a video server. An example of such a system is Xeoma (https://felenasoft.com/), which does not require installation and works under all popular operating systems: iOS and Android, Mac OS X, Linux ARM and Windows. The Xeoma program recognizes images well: faces, emotions, license plates and can work in a hospital ward, recreates 3D-objects and allows to draw maps for video surveillance points. The Xeoma window system is very primitive and resembles Windows. NT, and the blue-green pop-up settings windows have no inner edges. The windows have only one icon for closing. The confirmation buttons are darkened with white text and rounded edges.

Another program Zoneminder (https://zoneminder.com/) can work from any browser and there is a compilation for Linux with open-source code with the ability to study and edit the code for the tasks of any company. Zoneminder can detect movement in video and provides sensitivity settings, sends messages to a smartphone and/or email. The system is oriented for home use, can recognize images. The window system resembles GNOME or KDE in style, depending on which version of Ubuntu it is installed. Pop-up windows resemble tables with editable fields and blue inscriptions. Settings are allowed in the same window with the video image. Work is possible through Mozilla Firefox.

iSpy program (https://www.ispyconnect.com/) works only on Android, iOS and Windows. iSpy can recognize barcodes, license plates and overlay text on a video stream. The program has an open-source code, and it can connect an unlimited number of video cameras and video files. There is support for motion sensors, support for network broadcasting on YouTube and support of notifications. The window system resembles Windows 7, and the main menu of the application resembles Windows XP and older versions of Windows. There are large buttons in the menu. The screen is divided into blue sectors with settings. Pop-up windows and viewing video cameras with settings are made in the standard Windows 7 style.

Sighthound Video program (https://www.sighthound.com/) works under all operating systems except Linux. It is possible to work only with one video source. Sighthound Video has motion sensors and distinguishes a person from an animal. The Sighthound video surveillance system is aimed for country houses and can be integrated into the

smart home. The system works well with IP video cameras. The appearance of the windows is designed in the standard Macintosh style Sequoia 15.2, on the left side there is an area with certain fragments, and indicators are located above the video viewing, which occupies the main part of the screen. There is a chronometer under the video.

Another program AtHome Video Streamer (https://www.ichano.com/) can work with all popular operating systems. It works well with cloud services and can save video fragments on a schedule. AtHome can work with different types of devices: SmartTV, USB, IP, iOS, which is an undeniable advantage over other programs. The window system is very bright yellow, red with rounded buttons of the same color. At the top and left there is a light red menu with large gray buttons and bookmarks from video cameras (four per tab). At the bottom of the four videos there is a choice of viewing angles (three in total). The entire program window can only be minimized to the taskbar and closed, but it is not possible to open it on full screen.

EyeLine Program Video Surveillance (https://www.nchsoftware.com/) has the advantage of supporting more than 100 channels from different devices at the same time. The EyeLine program can send video to FTP and send SMS, has an archive manager and can send messages by e-mail. The window system has a Windows XP SP1 style with a large red "Stop" button on the video. On the left side of the video there is a directory with a mini view of five video cameras. At the top of the video is a menu for adding video cameras, settings and searching for saved fragments. There is a main menu of the application in the style of old versions of Windows (98/ME).

Netcam Studio program (https://www.netcamstudio.com/) can send notifications and can work with two video cameras. Netcam supports watermarks and consists of two components: the server part and the client part for Android, iOS and Windows 10. The program can work with analog video and combine recording with IP video cameras. There is support for audio signals and sound processing by neural networks. The window system is vaguely reminiscent of Windows NT with a design in gray tones. The menu at the top is made in the form of buttons with pictures and bookmarks (which are named as objects: home, work, office, workshop, warehouse, etc.). Each tab contains video cameras with a preview. At the bottom of the video there is statistical information. At the top of the window, it is possible to select the date and time of viewing, as well as update and preview the picture. At the top, there is also a menu for adding video cameras to tabs and searching in the library of the program that contains objects that should be identified. The settings can be defined on a separate tab. There is detailed information on the objects on each video camera. Pop-up windows are made in gray and instead of buttons there is a check mark and a cross to confirm or cancel. It is possible to configure the parameters of each video camera in the pop-up windows (manufacturer, model, resolution, IP-address, HTTP-port, login/password) and check the connection. The mobile application has two large buttons for recording and turning on the sound, and two switches for detecting motion and audio events. The main program for the PC has an event log. It is possible to work from a browser, but authorization is required (https://localhost:8100). There are indicators of server and client load, number of connections and amount of free space on the video server HDD. There is automatic synchronization with the cloud, remote control, connection indicator (at the bottom of the main window).

The considered systems have different capabilities. XProtect provides basic features in the Go version, but the features become much more with the acquisition of advanced versions of XProtect. XProtect is slow and requires installation, while Xeoma works without installation and runs very fast on weak computers. Without the program, they only work under Windows, but Zoneminder can also work under Linux. iSpy can work with mobile devices. Sighthound Video runs on Linux and Mac OS. At Home Video Streamer can work with devices. EyeLine runs on older versions of Windows. Netcam Studio works through a browser.

Comparison of the features of the considered systems is presented in Table 1. The systems are compared using the following parameters: p1 – operation of the system without installation; p2 – work on Linux; p3 – work with mobile devices; p4 – work on macOS; p5 – work with smart home devices; p6 – work with older versions of Windows; p7 – work in the browser; p8 – high speed of visualization; p9 – mobility; p10 – configurability; p11 – universality.

Table 1. Comparison of the considered features and limitations of the considered systems.

Name of the system	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
XProtect	−	+	−	+	−	−	+	−	−	+	−
Xeoma	+	+	−	+	−	−	−	−	−	−	+
Zoneminder	−	+	−	−	−	−	−	−	−	+	+
iSpy	−	+	+	+	−	−	−	+	+	+	+
Sighthound Video	−	+	−	+	−	−	−	+	−	+	+
AtHome Video Streamer	−	−	−	−	+	−	−	+	+	+	−
EyeLine Video Surveillance	−	−	−	−	−	+	−	+	−	+	−
Netcam Studio	−	+	+	−	−	−	+	+	+	+	+

According to the results of analyzes of existing systems (Xprotect, Xeoma, Zoneminder, iSpy, Sighthound Video, AtHome Video Streamer, EyeLine Video Surveillance, Netcam Studio) there is no system that meets all the parameters.

3 Problem Definition

Reasons why window systems are created in the browser to visualize events and video server settings:

1) Faster loading and operation of a windowed web site on slow and technological computers (embedded systems) where the standard Explore task from Microsoft works slow.
2) Support for drivers of complex technological devices and monitors.
3) Custom applications based on widgets, function graphs, complex tables, tooltips, running lines and mini applications [3, 4, 16].

The user's preferences are taken from the activity of his profile. It is possible to use the user's interests by IP address (open HTML pages, like Google ads) and other service information to train the neural network [19–22]. The option to use personal information is commonly proposed when installing the program (in the license agreement). The received contextual information can be used to create windows of the intelligent video surveillance system.

The browser's window system should be characterized by high speed of work, mobility, configurability and universality. It should be possible to open the window system on the KENSHI H10 LTE 4 GB RAM tablet with a resolution of 1280x800 (and a 10.1" diagonal) under Android 13.x, and on the MacBook Air M1 8 GB RAM with the macOS Sequoia 15.2 operating system in the Safari 18.2 browser, and on the VivoBook Asus R522M 4 GB RAM under Windows 11. The window system must work equally well as a primary task and a secondary task, like a webpage in a browser. The application must issue events and allow make settings on the video server under all modern operating systems and devices with a wide display [23–25]. It must be configurable depending on the user's preferences.

It is impossible to use the standard UNIX or Linux environment, because the computers have hardware adaptations for macOS and the standard UNIX and Linux kernels do not support Apple chips (M1, M2, M3, M4). Apple chips have non-standard processor registers, a different LLVM linker (https://llvm.org/), which can only work with Apple hardware. In addition, UNIX and Linux are overloaded with redundant functions that are not needed for a specific task (for example, for event visualization). By excluding additional functions, it is possible to increase the speed of the application (window system). In addition, the window system in the browser is well portable to other operating systems: QNX 6 Neutrino and QNX Software Development Platform 8.0 (https://blackberry.qnx.com/en), Debian Linux 12.10 (https://www.debian.org/), Open Solaris 11.4 SRU72 (https://www.oracle.com/solaris/solaris11/) and other less popular systems.

The results of the analysis of analogs showed that they lack mobility, speed, versatility and configurability. This can be achieved only if the window system is used instead of the standard Explore task. It will allow to ensures high speed and also mobility and versatility since the browser can be used on all devices and operating systems. All three criteria directly affect the speed of personnel interaction when solving applied tasks.

4 Solution Method

The authors of the article propose creating a website that imitates a window system. The website contains a status bar, a drop-down menu, and a window factory created in the Windows XP SP 3 style with the ability to move a window when hovering over the title (function «Drage-and-Drop»). The window contains icons for closing (cross) and centering or positioning the window in the center of the screen vertically and horizontally (circle). Windows are configured depending on the user's preferences. The user receives information that is a priority for him/her (an object entered the door; a dog ran along the playground). Binary incompatibility (Fig. 1) can be solved by creating a window system in the browser. First, the kernel functions are loaded, then the drivers, the graphical environment functions, and then not the application is launched, but the browser, which

connects instead of the application to the video server in the fog and visualizes the events and settings in the window system.

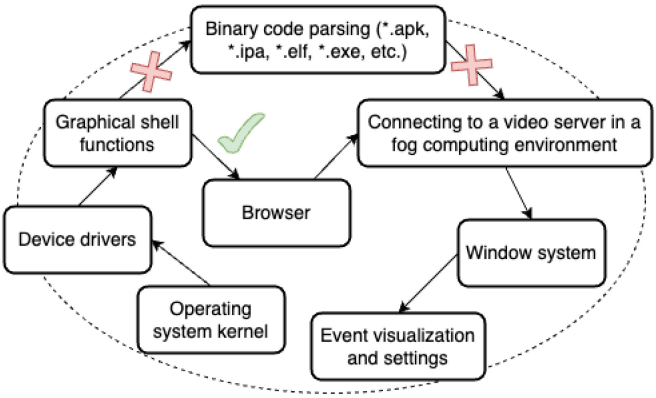

Fig. 1. Solution to the indicated problems in the browser.

Icons (Fig. 2) are located along the edges of the window title. The title and icons change color when hovered over, and when dragging the window, they change transparency to accurately determine the location on the screen. It is enough to use JS for development v.1.5 without «Node.js» on the client side with the formation of a tree structure DOM (Document Object Model) from basic elements in HTML/CSS. Dynamic queries to the database «MariaDB-v.10.4.28» are created through «iFrame», without AJAX technology and basic jQuery technology, because of which the website works much faster. The script «win.js» is built in such a way that the number of child windows is not limited. New windows are created at the «factory» in the file «index.html», where each DIV with the class «WinSettingsSpace» is a new window, regardless of whether it is written in the file «index.html» or created using JS-v.1.71 from the file «win.js». The CSS-v.2.0 properties of the DOM object are the same in both the "*.html" file and the "*.css" file and have event handling from the method of any object of the hierarchical structure. An example of an event handler when pressing the mouse button: «addEventListener ('click', function () {CreateWindow (3, 'Event statistics');});». JS technology is very easy to learn and is considered to have the next level in complexity after HTML/XML and HTML-tag properties with CSS styles.

Processing of dynamic requests (working with DBMS, uploading and processing files on the video server in fog) is done through the PHP language, since it is very fast in execution and easy to understand, with good and reliable documentation (https://www.php.net/docs.php). In the child frame of the "iFrame" type, the "dync.php" page opens, which is a PHP script with parameters. A typical example of such a script (Fig. 3) receives an array of data from one table "subEvents", but at the beginning creates a connection through the PDO object with parameters for connecting to the database "$pdoSet", switches to the "WinSubs" database. In case of an error in the "try… Catch" block, it issues an error message and terminates the work through the "die ();" method. Next, there is code for forming a query to the "subEvents" table. The final two-dimensional

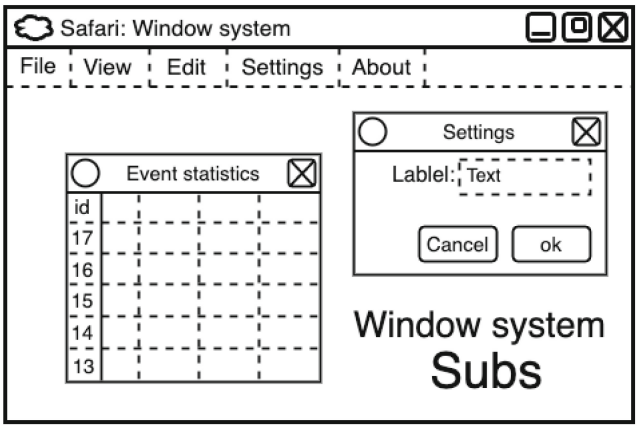

Fig. 2. A prototype of a developed window system named WinSubs-v.4.98a.

array "$resultMF[][]" of rows and columns contains only numeric indices, thanks to the "PDO::FETCH_NUM" modifier. Then the DOCTYPE header is formed in HTML5 format, after opening and closing the PHP code (" < ?php … ? > "). Next, the table is formed in a loop iterating over a two-dimensional array. The table has an "id" tag with the value "iTableGet" and the content is passed from the child frame to the main page "index.html", where the first frame in the DOM document was assigned to the "iframe" variable via the global object "document" of the method "getElementsByTagName ('iframe')[0]" and was called via the modifier "iframe.src = './dync.php?event';". Instead of PHP, one can use another programming language, for example, SubScript [2].

5 Case Study

The authors of the study developed a website that runs entirely on JavaScript 1.7, implementing a window system in the browser. In the center (Fig. 4) of the website there are two windows. The window at the top shows the version of the website, and the second window at the bottom of the screen visualizes events. At the top of the screen there is a status bar, reminiscent of macOS or the main menu of the application (a form element in the Windows application designer). In the upper left corner additional information about the detected event is shown in a small sub-window. Sub-windows are created according to the factory principle, very quickly, almost instantly. Visualization of events is based on using fog distributed computing and containers or virtual machines. In this example, the Parallels Desktop for Mac 20.0.0 (https://www.parallels.com/products/desktop/) virtual machine type based on Qt Creator version 6.0.2 (https://www.qt.io/product/development-tools) was used.

An important element of the proposed solution is the possibility to take into account the user's preferences, which are configured in additional windows that can be opened through the main menu of the application in the browser. It allows to set the color, size, position of windows and much more using a neural network or all these settings can be changed manually using buttons, checkboxes, drop-down lists, etc.

```
<?php
if (isset($_GET['event']))
        try {
                $pdoSet = new PDO('mysql:dbname=test;host=localhost','root','');
                $pdoSet ->query('SET NAMES utf8;USE WinSubs;');
        } catch (PDOException $e) {
                print "Error!:".$e->getMessage ()."< br />";
                die();
        }
        $sql = "SELECT * FROM subEvents WHERE id>=498 ORDER BY id DESC";
        $stmt =$pdoSet ->query($sql);
        $resultMF = $stmt -> fetchAll (PDO::FETCH_NUM);
        if (Count($resultMF) == 0) {
                ?>
        <!DOCTYPE html><html><head></head><body>
<div id=' iTableGet '>none</div></body></html>
        <?php
        } else {
                $sHTML = '<table>';
                for ($i = 0; $i < Count($resultMF); ++$i) {
                        $sHTML.='<tr>';
                        for ($iCol = 0; $iCol < Count($resultMF [$i]);  ++$iCol)
                                $sHTML.= '<td>'.$resultMF [$i][$iCol].'</td>';
                        $sHTML.='</tr>';
                }
                $sHTML.='</table>';
                GetHTML($sHTML, 'good');
        }
}
?>
```

Fig. 3. PHP code for dynamic loading of information (downloading one table from the DBMS and transferring it to the main frame).

The developed window system was tested in the St. Petersburg Subway (https://metro.spb.ru/). The neural network for detecting events that occur in the subway, such as broken escalator steps, was trained using dataset of 34812 images and a control dataset of 873 images. The dataset contains the images of the following classes: broken steps, fire, flooding, etc. The architecture of the neural network based on NumPy 2.2.0 (https://numpy.org/) with 14 layers: subsampling, max-pooling, and convolution was used.

The features of the NumPy algorithm are: homogeneity of the data array elements, multidimensionality of NumPy arrays with an unlimited number of changes, fixed array dimension after its creation, efficiency in terms of simultaneous operations on the data array, a wide choice of convenient methods for working with array data. NumPy tools work very quickly and have a higher data processing speed in «C/C ++» than in Python. There are no problems with training using the NumPy algorithm in the process of machine

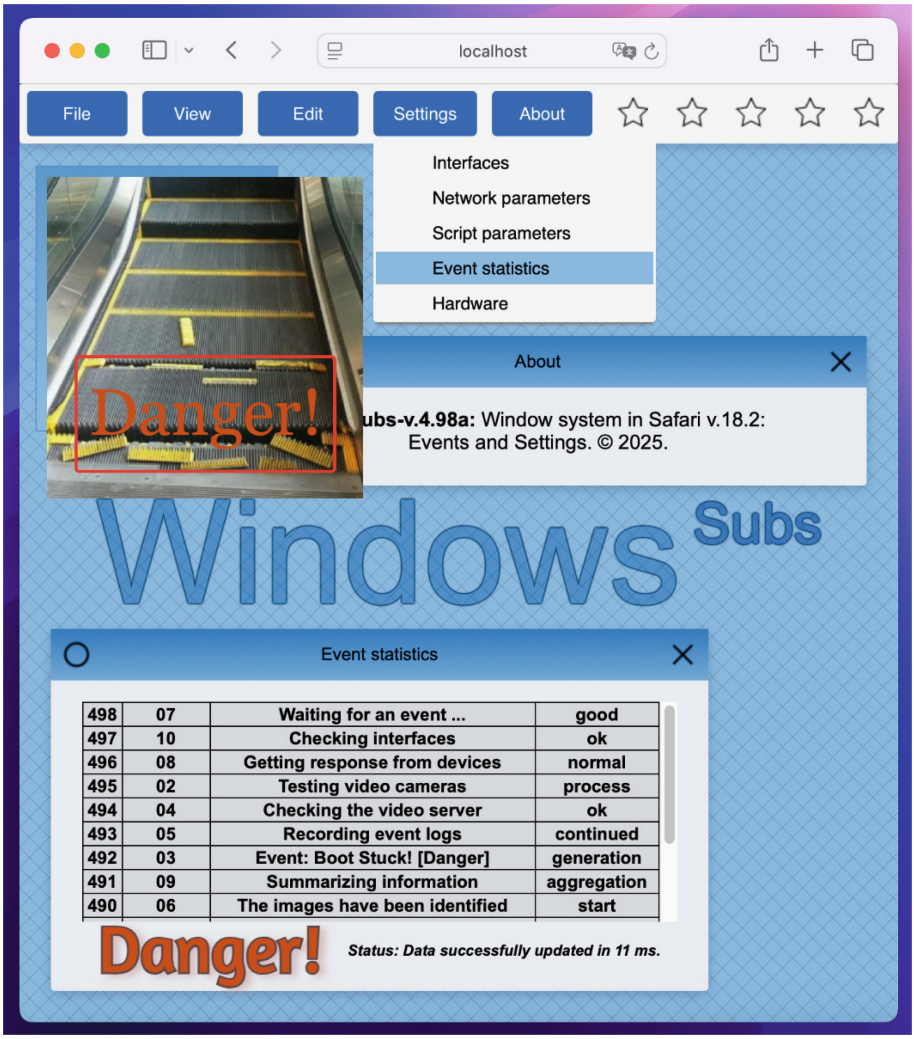

Fig. 4. Windowing system in Safari v.18.2 browser for settings and visualization of events.

learning, since NumPy is implemented very well. A dataset with images that were used to train the model can serve as a visual graphical material. To examine the images carefully, just download the project, run the application build and the datasets will be loaded into the project on NumPy 2.2 in the directory with the source codes. The following image processing libraries were used in the process of using the NumPy 2.2.0 script: Scikit-image 0.25.2 (https://scikit-image.org/), OpenCV 4.11.0 (https://opencv.org/), Mahotas 1.4.16 (https://mahotas.readthedocs.io/).

The neural network model was trained to identify patterns during 12 h on NVIDIA GeForce RTX 4090 (https://www.nvidia.com/en-eu/geforce/graphics-cards/40-series/rtx-4090/). WinSubs window system classified correctly more than 92% of images and

showed high efficiency which was estimated based on 5 parameters (p1-p5) shown in Table 2: p1 – fire service; p2 – ambulance (urgent service); p3 – water supply and sanitation; p4 – subway police; p5 – train repair and track workers. The overall increase in the speed of reaction on detected events has increased by 20.3%.

Table 2. Comparison of window systems by response time of various services (p1-p5) in the subway upon occurrence of an event (in sec.).

Window system	p1	p2	p3	p4	p5	More than WinSubs (%)
Explore	137	251	893	192	1739	30.3
XProtect[a]	132	241	857	189	1548	**20.3**
Xeoma	148	259	942	193	1687	30.9
Zoneminder	151	273	1019	214	1819	41
iSpy	147	256	938	185	1635	28.2
Sighthound Video	161	284	982	201	1719	35.7
AtHome Video Streamer	157	278	1124	224	1842	47
EyeLine Video Surveillance	159	287	1207	231	1947	55.4
Netcam Studio	143	251	904	192	1629	26.5
WinSubs	116	219	691	147	1293	0

a) The default system in the subway (now).

The bar chart created based on the experimental data visually illustrates (Fig. 5) how faster WinSubs is (%) compared to other visualization and event monitoring systems. The slowest compared to WinSubs was EyeLine Video Surveillance (55.4%), and the real competition is XProtect (20.3%). Netcam Studio and iSpy programs are at approximately the same level (26–29%). Xeoma and Explore programs are also at approximately the same level (30–31%). After the EyeLine Video Surveillance program (55.4%), the second place for the slowest operation is taken by AtHome Video Streamer (47%), perhaps due to the adaptation of the program for home use.

To reproduce the experiment, it is necessary to install the WinSubs-v.4.98a window system (https://github.com/alex1543/wincat) instead of the desktop (disable the Explore Windows x64 desktop via the "Regedit.exe" utility in the directory "C:\Windows\System32", delete the startup parameter for the Explore task). Collect statistics on one object, where there are more than 159 Axis video cameras P13, P33 and P1365MKII (https://www.axis.com/). Technical specifications of the computer for collecting statistics: ASUS VivoBook R522M with 4 GB RAM, 128 GB ROM under Windows 11.

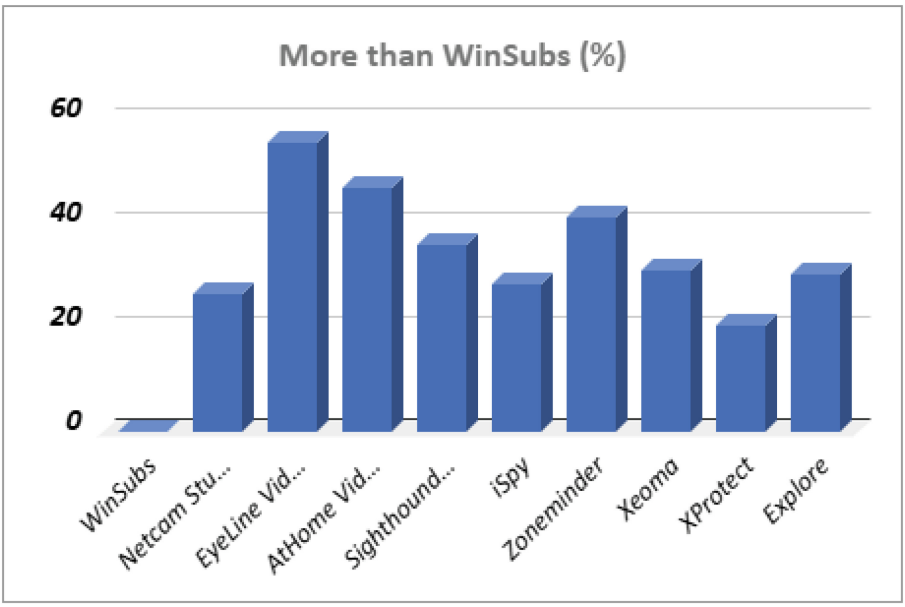

Fig. 5. Visual representation of how much faster WinSubs is (%) than others.

6 Conclusion

The authors of the article consider the creation of a window system in a browser instead of the standard Windows desktop. The key advantage of the proposed system is the high speed of window system rendering. Another distinctive feature of the new WinSubs window system is the versatility, since it can work on all devices and operating systems. In addition, the system is mobile and easily portable, so it can be launched by a video surveillance operator on a tablet. The program is easily configured.

Event visualization based on distributed fog computing has made it possible to significantly increase the speed of the window system in the browser and improve the performance of services.

The usage of the developed system in subway showed an increase in the efficiency (speed of performance) of personnel by 20.3%.

The scientific component of the study is in the method for solving a wide range of problems where it is necessary to determine events in systems with limited resources or non-standard hardware.

Load testing of the system was performed on a control dataset for two weeks with 47 Axis P3384-V IP-cameras (https://www.axis.com/) with positive results and the system did not show any operational failures.

In future it is planned to use a neural network with controlled elements (RNN-CE network), which will allow consider an extended list of parameters such as color, location on the screen, window size, frame event (a person entered the door) and many other for adaptive dynamic configuration of the window system according to the user's preferences. The choice fell on this neural network, since it supports additional training

in dynamics and the neural network algorithm works well in recognizing images in motion [5]. Such neural network will be deployed on the video server in the next version of the program.

The proposed approach can be applied not only in the subway, but also in other areas of limited space: a cruise liner, a submarine, a school, a kindergarten, a shopping center, a wholesale warehouse, where it is necessary to conduct investigations and monitoring.

Acknowledgments. This work was supported by the state budget, project No. FFZF-2025–0019.

References

1. Tanenbaum, Andrew S., Woodhull, A.S.: Operating Systems: Design and Implementation. ISBN 9780131429383, January 2006
2. Subbotin, A., Delhibabu, R., Zhukova, N.: A script language creation for event visualization and data storage in an intelligent video surveillance system. Adv. Artif. Intell. Mach. Learn. **4**(4), 3161–3172 (2024)
3. Osipov, V., Zhukova, N., Subbotin, A., et al.: Intelligent escalator passenger safety management. Sci. Rep. **12**, 5506 (2022). https://doi.org/10.1038/s41598-022-09498-x
4. Tianxing, M., Vodyaho, A., Zhukova, N., Subbotin, A., et al.: Urban intelligent assistant on the example of the escalator passenger safety management at the subway stations. Sci. Rep. **13**, 15914 (2023). https://doi.org/10.1038/s41598-023-42535-x
5. Osipov, V., Nikiforov, V.: Recurrent neural networks with controlled elements in restoring frame flows. Inf. Control Syst., 10–17 (2019). https://doi.org/10.31799/1684-8853-2019-5-10-17
6. Yokoyama, N., Tanaka, K.: Container auto-scaling system using sliding-window regression with fuzzy entropy. J. Inf. Process. **32**, 916–928 (2024). https://doi.org/10.2197/ipsjjip.32.916
7. Liu, Y., Yanjun, F.: Design of the nonlinear system predictor driven by the bayesian-gaussian neural network of sliding window data. Comput. Inf. Sci. **2**, 26 (2024). https://doi.org/10.5539/cis.v2n2p26
8. Liang, Z.: Anti-pollution intelligent window control system based on air detection. J. Phys. Conf. Ser. **2810**, 012004 (2024). https://doi.org/10.1088/1742-6596/2810/1/012004
9. Ko, W., Burgess, I., Chung, S., Macnaughton, P., Um, C.Y.: Assessing the impact of glazing and window shade systems on view clarity. Sci. Rep. 14 (2024). https://doi.org/10.1038/s41598-024-69026-x
10. Sisi, C.: Empirical study on functional defects in multi-window Android systems. Appl. Comput. Eng. **73**, 288–294 (2024). https://doi.org/10.54254/2755-2721/73/20240413
11. Koay, H.-H., Thien, G., Wen Cheun, B., Hamed, A., Ruthramurthy, B., Ng, Z., Chan, K.: An IoT-enabled monitoring and control system for electrochromic smart windows. Discover Internet Things 4 (2024). https://doi.org/10.1007/s43926-024-00086-1
12. Deng, L., Shi, Z., Wang, Y., Yu, X., Guan, Y., Xu, Z.: Hybrid window decoding for joint source channel anytime coding system. Entropy **26**, 940 (2024). https://doi.org/10.3390/e26110940
13. Wang, F.-C., Huang, H.-T.: Extended-window algorithms for model prediction applied to hybrid power systems. Technologies. **12**, 6 (2024). https://doi.org/10.3390/technologies12010006
14. Febriyanti, R., Ranti, B., Shihab, M., Wicaksana, I., Mekkawati, H., Salbari, F., Diarto, N.: Transformation of business licensing through a single submission system on public service efficiency: a case study of the Indonesia National Single Window Agency (LNSW). Indonesian J. Comput. Sci. 13 (2024). https://doi.org/10.33022/ijcs.v13i5.4411

15. Wang, A.: Engineering design and implementation of AI-driven single window systems for international trade. J. AI-Driven Trade Facilitation Eng. Single Window Syst. **1**, 21–34 (2024). https://doi.org/10.6914/sw.010102
16. Gunasegran, S., et al.: IoT-based smart window and temperature control system for optimized thermal comfort. 020018 (2024). https://doi.org/10.1063/5.0230083
17. Subbotin, A., Zhukova, N., Gudilov, M.: Problems of building digital twins of escalators at subway stations based on machine learning. In: Arai, K. (eds.) Intelligent Computing. SAI 2024. Lecture Notes in Networks and Systems, vol. 1018. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-62269-4_13
18. Zhang, W., et al.: Design and development of a new biped robotic system for exoskeleton-structure window cleaning. IEEE Trans. Automation Sci. Eng. PP, 1–12 (2024). https://doi.org/10.1109/TASE.2024.3390030
19. Liu, L., Li, F., Liu, W., Xia, H.: Sliding window iterative identification for nonlinear closed-loop systems based on the maximum likelihood principle. Int. J. Robust Nonlinear Control **35** (2024). https://doi.org/10.1002/rnc.7705
20. Zhou, Q.: Frequency-domain characterization of finite sample linear systems with uniform window inputs. Signals. **6**. 1 (2025). https://doi.org/10.3390/signals6010001
21. Pietzsch, A., Woerd, J., Andrae, M., Gebbeken, N.: Analysis of the load-bearing behavior of catcher-cable systems for hazard prevention for windows under blast loads (2024)
22. Xiao, H., Peng, H., Zhao, Y., Hou, L.: Magnetic field energy harvesting and power management with transfer window modulation. IEEE Trans. Power Electron. PP, 1–13. https://doi.org/10.1109/TPEL.2025.3528110
23. Tung, T.-J., Al-Hussein, M., Martinez, P.: Vision-based guiding system for autonomous robotic corner cleaning of window frames. Buildings **13**, 2990 (2023). https://doi.org/10.3390/buildings13122990
24. Zhao, F., et al.: Continuous solar energy conversion windows integrating zinc anode-based electrochromic device and IoT system. Adv. Mater. **36**, e2405035 (2024). https://doi.org/10.1002/adma.202405035
25. Gao, L., Zong, R.: Design of intelligent window automatic monitoring system based on microcontroller control. J. Electron. Res. Appl. **8**. 34–40 (2024). https://doi.org/10.26689/jera.v8i4.7922