Adv Syst Sci Appl 2024; **03**; 73-89
Published online at https://ijassa.ipu.ru.

The Introduction of Cloud Storage as One of the Ways to Save the Organization's Data Management Resource

Dmitriy Kornienko[1*], Alexander Nikulin[1], Nikita Kornienko[2]

[1]*Bunin Yelets State University, Yelets, Russia*

[2]*National Research University Higher School of Economics, Moscow, Russia*

Abstract: The article discusses the use of cloud technologies in enterprise data management, provides examples of the use of cloud technologies. The analysis of the problems of implementation and functioning of cloud data storages in organizations has been carried out. Based on the analysis, the technological and functional problems of implementing cloud data storages are highlighted. The attention is focused on unsolvable problems of building distributed data transmission systems. The practical implementation of cloud storage and data exchange processes through it has been carried out. A comparative analysis of the proposed methods with existing ones is demonstrated and their effectiveness is proved.

Keywords: cloud technologies, data, distributed storage system, protocols.

1. INTRODUCTION

Improving information technology occupies an important place among the many new directions in the development of organizations. It is aimed at the development of the information environment, which involves the introduction and effective use of new information services. One of the promising directions of development of modern information technologies is cloud technologies. Cloud technologies are understood as technologies of distributed data processing, in which computer resources and capacities are provided to the user as an Internet service. Let us analyze the essence and main characteristics of cloud technologies in order to substantiate the possibility and expediency of their application in organizations. Cloud computing is a model for providing ubiquitous and convenient network access (on an as-needed basis) to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be quickly provisioned and released with minimal management effort and need interaction with the service provider. In cloud computing, data is permanently stored on virtual servers located in the cloud, and is also temporarily cached on the client side on computers, laptops, netbooks, mobile devices, etc. One of three basic models is used to build a cloud: software as a service, platform as a service, infrastructure as a service. The cloud is not the Internet itself, but the entire set of hardware and software that ensures the processing and execution of client requests. Cloud computing is a new paradigm that involves distributed and remote processing and storage of data. The essence of cloud technologies is as follows: you can not have any programs on your computer, but only have access to the Internet. Remote access to data in the cloud can be found anywhere on the planet where there is access to the Internet.

A cloud data storage system, or data storage as a service is an abstract concept that corresponds to a data storage system, which can be administered on demand using a special interface [1-3]. This interface abstracts the location of the system, so it does not matter whether it is local, remote, or hybrid. Cloud storage infrastructures form new architectures that support different levels of service on top of a potentially large group of users and

* Corresponding author: dmkornienko@mail.ru

geographically distributed storage devices. It is important that clients can control and manage how their data are stored as well as how their money is spent. Numerous cloud service providers offer management tools that provide users with enhanced control over costs.

Storage efficiency is an important characteristic of a cloud storage infrastructure, especially given its emphasis on overall savings. To make the storage system more efficient, you need to store more data. A common solution is to reduce the amount of source data so that they occupy less physical space. There are two ways to achieve this goal: compression, i.e., packaging data by encoding them using different representations and deduplication, i.e., excluding all duplicate data [4; 5]. Although both methods are useful, compression involves processing (transcoding data to and from the infrastructure), and deduplication involves calculating signatures to search for duplicates.

Saving is one of the most important features of cloud data storage. This involves purchasing storage devices, their power supply, repair, as well as storage management. Considering cloud storage from this standpoint (including a Service Level Agreement and the increased storage efficiency), it can be beneficial for certain usage models. A storage access API (application programming interface) is an essential component of the services. Many applications require access to storage services using an API that is optimised for a particular storage system, either on their own hardware or cloud-based. The Amazon S3 API cloud storage system provides developers with an SDK (software development kit) for .NET and Java, as well as libraries for additional platforms and languages. These interfaces usually use Representational State Transfer (REST) protocols and/or Simple Object Access Protocol (SOAP) [6-8].

When considering the architecture, it is necessary to take into account its operating parameters. They are understood as various characteristics of the architecture, taking into account the cost, performance, remote access, etc. The architecture of cloud data storage is primarily the delivery of data storage resources on-demand in a highly scalable and multi-agent environment. In general, the cloud storage architecture is an external interface that provides an API to access storage devices. In traditional storage systems, this is the SCSI (Small Computer System Interface) protocol, but the cloud requires new ones. Among them, you can find external protocols of Web services, file protocols and even more traditional external interfaces (Internet SCSI, iSCSI, etc.). Middleware i.e. the data storage logic is behind the external interface. This middleware functions as data replication and data reduction, according to traditional data placement algorithms taking into account geographical location. Finally, the internal interface organises the physical storage of data. This can be an internal protocol that functions specifically or a traditional server with physical disks.

2. THEORETICAL OVERVIEW

The rapid increase in the bandwidth of computer networks allows the development of numerous applications that provide for intensive data processing [1]. These new applications can perform tasks ranging from mass data transmission (SDSS (Sloan Digital Sky Surve) and electronic Very Long Baseline Interferometry) to high-bandwidth interactive systems (GeoWall) [2]. However, different applications have different requirements for data transmission services [3]. For example, the GeoWall application may require a long-term speed variation of data transmission, whereas data should be transmitted at the maximum possible speed in private networks for the SDSS application [4]. However, the current Internet system is designed to provide support for a variety of different types of applications. This Internet design philosophy affects fundamentally the development of transport protocols. On the Internet, most of the traffic is generated by TCP (Transmission Control Protocol), but there are applications, which are not provided with a sufficient level of efficiency by TCP. In

the context of high-performance computing, TCP is well known for its low efficiency and network fair sharing with threshing [5; 6].

Modifications to the network stack of the protocol kernel (for example, new variants of TCP) usually require several years for standardisation, implementation and widespread deployment. Indeed, since the advent of TCP, about three decades ago, only four versions of this protocol have been widely deployed, namely Tahoe, Reno, NewReno, and SACK [7]. Although today the data transmission speed is 1 Gbit / s and higher for a growing number of networks, a wide bandwidth remains an urgent problem for web applications because of the limitations of existing network transport protocols. The limitations of the implemented network transport protocols are one of the main reasons why it is so difficult to scale applications with intensive use of network connections from local clusters to global networks [8].

The Transmission Control Protocol (TCP) has been successfully used for decades as the main transport layer protocol of the network protocol stack. However, it has recently been shown that TCP loses performance when used for high-speed Wide Area networks, especially for geographically remote networks. TCP uses the additional increase/multiplicative decrease congestion control algorithm, which cannot provide all the available bandwidth and has sufficiently large latency in the case of large packet loss in high-performance networks [9].

Computer network researchers labour at new transport protocols and congestion avoidance algorithms to support next-generation high-speed networks. Many studies, including the TCP variants (FAST, BiC, Scalable, and HighSpeed) and XCP showed higher performance when modelling them [10]. However, the practical use of these protocols is still very limited in real applications due to the difficulties of their deployment, installation and hardware restrictions [11]. Network users usually turn to application-level solutions when they need to transfer a large bulk of data, among which UDP-based (User Datagram Protocol) protocols are very popular, for example, SABUL, UDT (Data Transfer Protocol), Tsunami, RBUDP (Reliable Blast UDP), FOBS and GTP [12]. Such UDP-based protocols provide much better algorithm strength and are easy to install. However, despite the simplicity of deployment at the user level, the protocols are quite difficult to configure in the kernel to make them as efficient as possible [13]. Since the user-level deployment cannot change the kernel code, there may be additional context switches and copying process of memory sections between the user level and the kernel one. At high data transfer rates, these operations are very sensitive to processor load and protocol performance [14].

3. MATERIALS AND METHODS

For the practical application of the developed methods, it is necessary to design and develop a distributed data storage system. The specifics of modern data storage are in their transition to distributed systems. Therefore, a necessary condition is to ensure the redundancy of critical network nodes. It is also an important condition that network nodes have information about other nodes and their data. In the conditions of distributed operations, their necessary functionality is to ensure timely identification of an inoperable data exchange/storage node and their extraction from the operation. Also, after the node is restored, it should join the data exchange/storage process as soon as possible. Designing such a system requires a preliminary analysis of the requirements and planning its internal processes.

The software of the system must ensure complete functionality and have the means to organise all the required processes of processing, transmitting and storing data in all regulated modes of operation. The system software must be universal; functionally sufficient; reliable; adaptive; upgradeable and scalable; have an intuitive user interface; protected from external influences; able to record all actions of software users. The software

should be developed using the principles of structural and modular programming. Each of the system tasks should be as independent as possible from the others.

Quality control of developed software tools should be provided by testing and conducting a trial operation. The system architecture should be based on the principles of high-load and fault-tolerant systems. It must meet the following basic requirements: the system should support total scaling-out (both data storage nodes and satellites to improve the efficiency of data access); the system should be able to duplicate all critical network nodes and data storage to ensure system fault tolerance.

The system should include the following functional components:

1. Data storage is a component that provides data storage for solving the following tasks: data storage; data access; accounting for user actions.

2. Satellite is a component that provides fast information and hardware interaction of the system with its users.

3. Adaptive DNS (Domain Name System) server is a component that provides logistics of requests from a user to the system.

General requirements for the system reliability:

1. The software and hardware complex should function around the clock, in a continuous mode, except for force majeure circumstances.

2. The databases should be backed up regularly (at least once a day). It is necessary to have at least two backups of all data. Backups should be stored in physically remote locations.

3. Failures and malfunctions of workstations and network devices should not result in data destruction and affect the performance of the whole system.

4. Failure of one of the subsystems should not shut down other subsystems, i.e., all other subsystems must function.

5. A planned stop or failure of the information resource should not lead to a software failure.

6. Incorrect user actions should not result in an emergency.

7. Errors of technical personnel should be minimised, including by clearly delineating access rights to the system, as well as logging system events.

The reliability of an information system is understood as the general system ability to preserve its main specified properties in time. With this understanding, the software should: be resistant to erroneous actions of users (errors in the actions of personnel should not lead to failures (failures) in the operation of the information system software); provide guaranteed control of incoming and outgoing information; ensure rapid recovery after failure (failures).

The software is developed on the basis of common operating systems, programming tools and DBMSs (database management systems). The system should use standard solutions based on standard protocols and interaction interfaces, which provide for the possibility of interfacing and collaboration of hardware and software from various manufacturers, as well as for interfacing with information systems of other organisations. All hardware and software solutions used in the system design must comply with the requirements of national standards or international standards (if applicable). The hardware and software used as part of the IS must be certified or otherwise documented by a supplier company confirming their compliance with the specifications.

Unified solutions are preferred due to the great social significance of the project, and the tight deadlines for commissioning. Such solutions should have the following properties: access to the system should be provided via the global Internet; modularity (component solution); be able to integrate with external automated systems. The system must provide the following functions: registration of system clients; creation of a company cloud record in the system; creation of an administrator account for the company; administrative part; registration of system users for the company; editing user profiles; providing access to data via protocols: HTTP (HyperText Transfer Protocol); HTTPS (HyperText Transfer Protocol

Secure); FTP (File Transfer Protocol); FTPS (File Transfer Protocol + SSL); SFTP (Secure File Transfer Protocol); error handling; storage errors; errors satellites; script errors; communication interruptions; heavy load; registration of user actions.

4. RESULTS AND DISCUSSION

Since the list of necessary functions and the relationships between them have been analysed, it is advisable to present them in the form of system use cases (Fig. 1, 2). The system should provide effective information exchange between internal components. Information exchange should be carried out between the system components using local computer networks and global data transmission networks [15; 16]. Appropriate specified information exchange protocols should determine the composition, structure, volume and frequency of transmission. Information exchange protocols should provide measures to exclude the possibility of unauthorised access to data.

There should also be means for monitoring the transmitted input/output data and means for monitoring information in databases. The requirements for information exchange between the system components should be determined at the development stage, based on the capabilities to deploy the platform [17; 18].

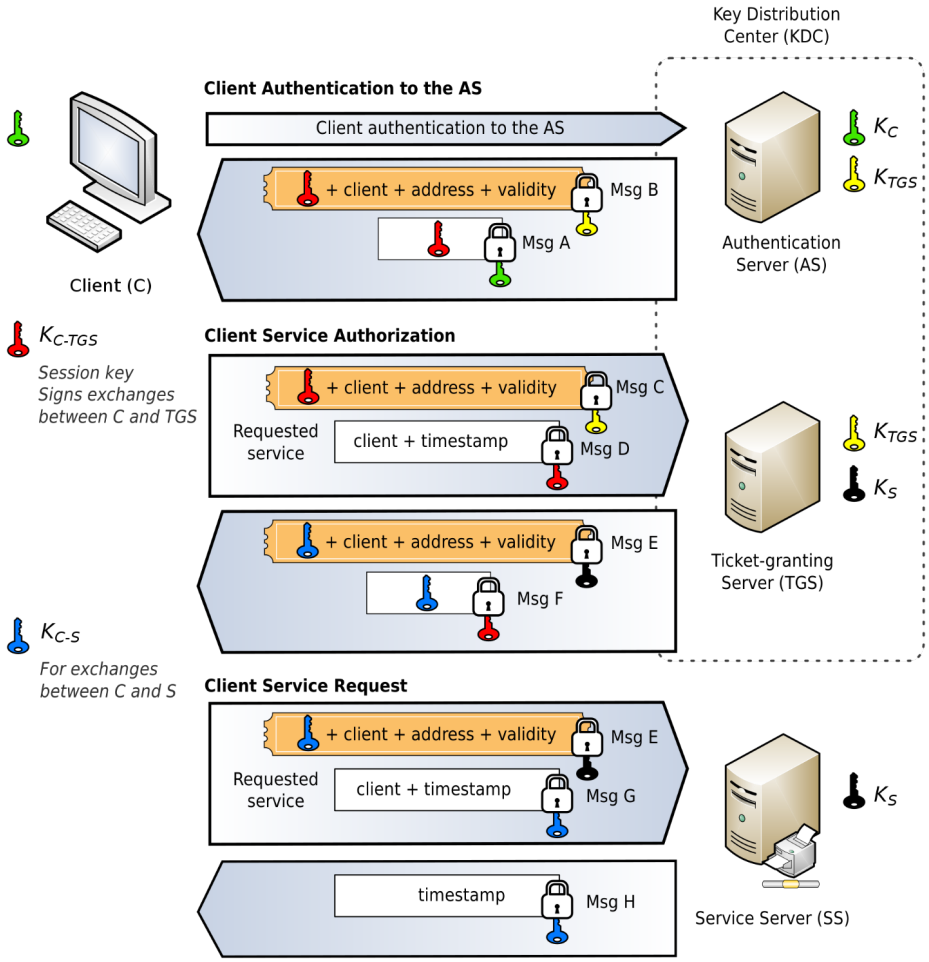

Fig. 1. Use Cases of the System from the Point of View of a User and a Client of the System

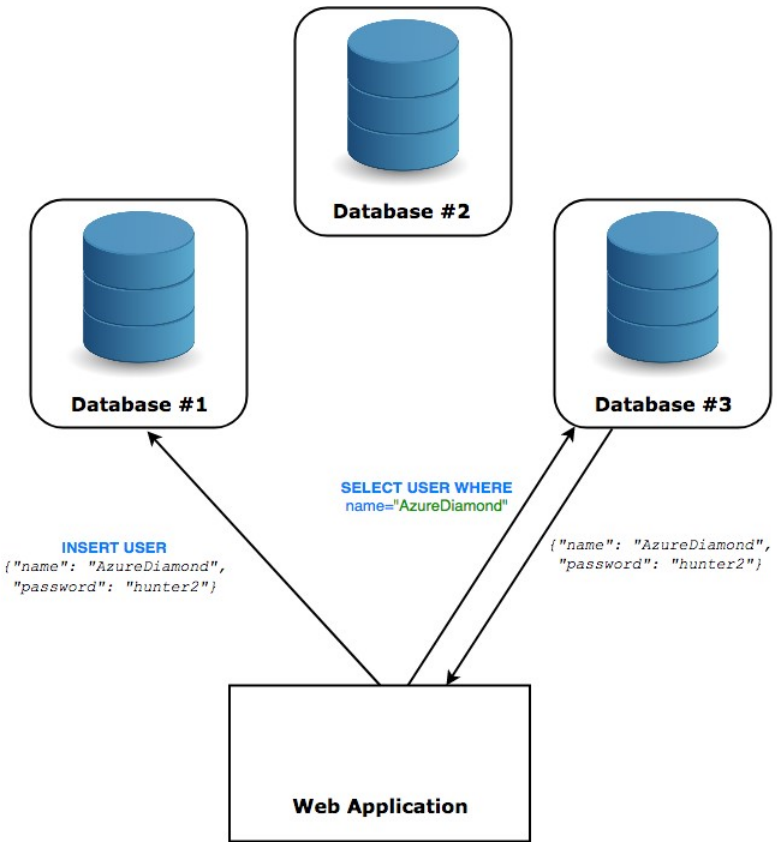

Fig. 2. Options for using the system from the point of view of the System Owner

The overall architecture of the system consists of an adaptive DNS server, several data containers (DCs) and many satellites (Fig. 3). Each data container should include (Fig. 4): two firewalls; two application servers; two storage units with disks. Such an architecture is necessary to ensure the fault tolerance of the system in case of loss of a physical network node [19; 20]. At the first stage, requests come to the firewall level, which is displayed with the same IP (Internet Protocol) address with indexes 0 and 1 for external nodes. If a device with 0 exits, then all requests are readdressed to a device with 1.

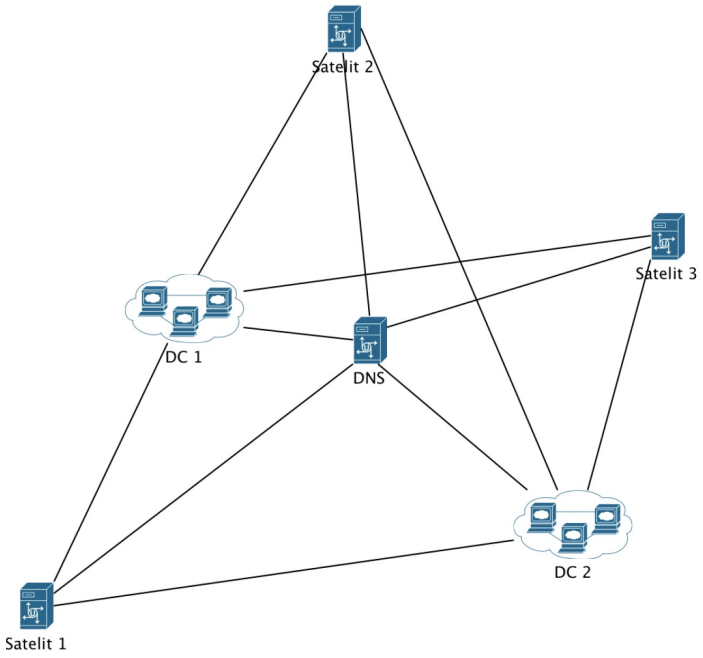

Fig. 3. General architecture of the system

After a firewall receives a user request, the request gets in turn to the available application server, the availability status of which is constantly monitored at the first level. After the hardware connection, the two storage units are connected at the hardware level as a single storage unit. Also, at the software level, data synchronisation is configured between the units to ensure data backup. Since the speed of processors and the amount of RAM increase extremely quickly, the application servers cannot make full use of them without complex mathematical algorithms and are limited by the number of users, therefore, it is advisable to use containers (virtualisation at the operating system level). Then, structurally, the application server will consist of several containers and a load balancer (Fig. 5).

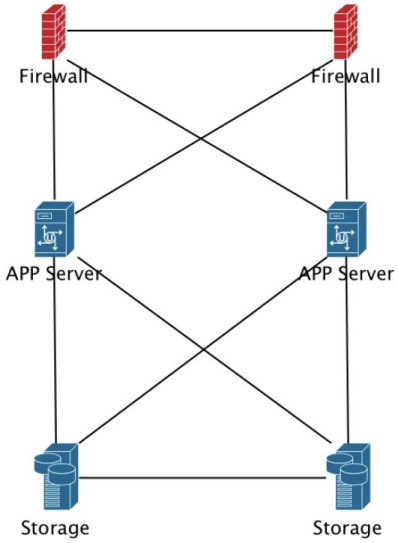

Fig. 4. Single Storage General Architecture

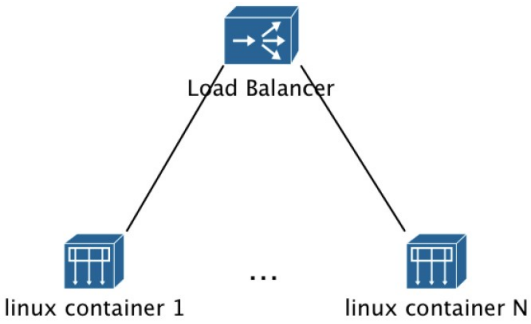

Fig. 5. Application Server General Structure

There are several software tools aimed at monitoring the availability of services in containers, and KeepAlived is the most popular of them. Their operational disadvantage is that they cannot group services when used in our architecture. Since the data accessibility of different users from different companies via FTP is one of the key requirements (each company has its own hostname), one needs different access IP address for each company (FTP specifics). An FTP server can support the simultaneous use of many IP addresses. In this case, KeepAlived will try to create connections and check availability for each IP address, and the load will increase exponentially on the server to check the availability of containers if there are a lot of them. To solve this problem, a system has been designed that completely replaces the KeepAlived one with the following features:

– grouping IP addresses and services as one service (in case of unavailability, this service ceases to be available at all specified IP. And vice versa, the services become available again at all IP addresses if available);

– storing statistics for all IP addresses and services, taking into account the number of requests; the data transmitted; the data received; inoperability; restoration of operability.

Architecturally, a satellite corresponds to the architecture of an application data storage server. It does not require a large and reliable data warehouse, since it acts exclusively as a "proxy server" between clients and their data. In addition, there is no need to completely duplicate it, since it does not contain critical data and all input requests will be automatically resent to other nearby satellites.

Modern cloud data warehouses must provide both the most rational data flow and a significant increase in its rate, i.e., acceleration of source-consumer data transmission and processing. First, data flows are analysed to solve these problems, when designing an information system:

– all the links of the data processing and storage system, starting with initial information, its gradual transformation and formation of final data sent to the managed system as a reporting command and other information. At the same time, the role is determined of each system element in the cloud storage and recorded in the data processing scheme, as well as their structure and functions are specified;

– a data communication diagram of all system elements between themselves and the external environment. The diagram may contain information about specific forms of data links and indicate their quantitative and temporal characteristics;

– find primary (output) system data. An information flow analysis is the most important stage in the deployment of an existing data storage system, which will meet the design objectives.

Studying information flows gives a general idea of how a system functions and is the first step in analysing the effective design of a highly loaded data processing, storage and transmission system. Further study of data flows allows identifying the elements of the information display of objects, their relationships, as well as the structure and dynamics of data flows. The movement of data, accompanied by appropriate information flows, is the basis for ensuring the operation of a cloud data warehouse in the system. There is a continuous movement of information flows between all the elements of the system functioning in the data warehouse environment, that provide the receipt of information data necessary for the analysis and transmission of client data. The main elements of the system are (Fig. 6):

1) a data warehouse, which is one of the main elements of the system, where client data are physically and long-term stored (Fig. 7);

2) A data access satellite, which is a data access element in the system that is located as close as possible to users (data consumers), and may be used as temporary storage (caching) (Fig. 8);

3) A DNS server is an element of the system that receives dynamic updates from its other elements and provides data users with reliable and up-to-date information.

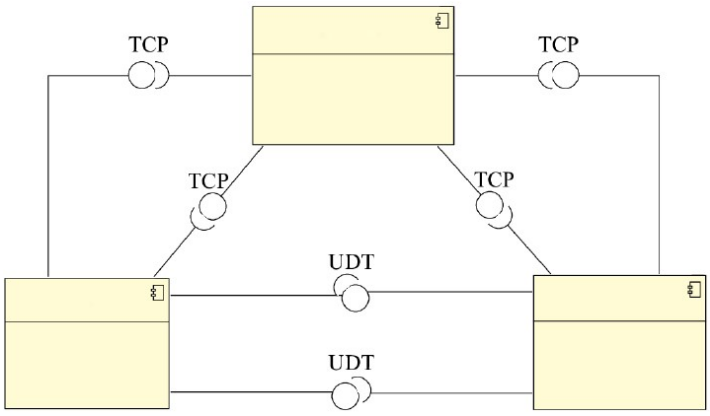

Fig. 6. Main System Elements Interaction Diagram

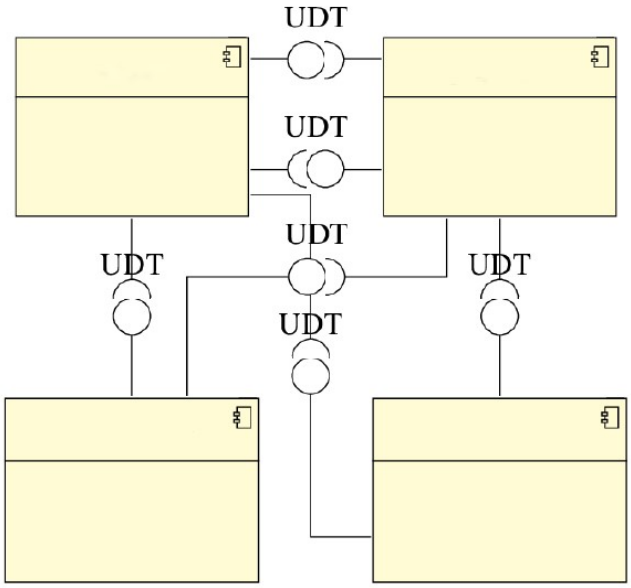

Fig. 7. Satellite Deployment Diagram

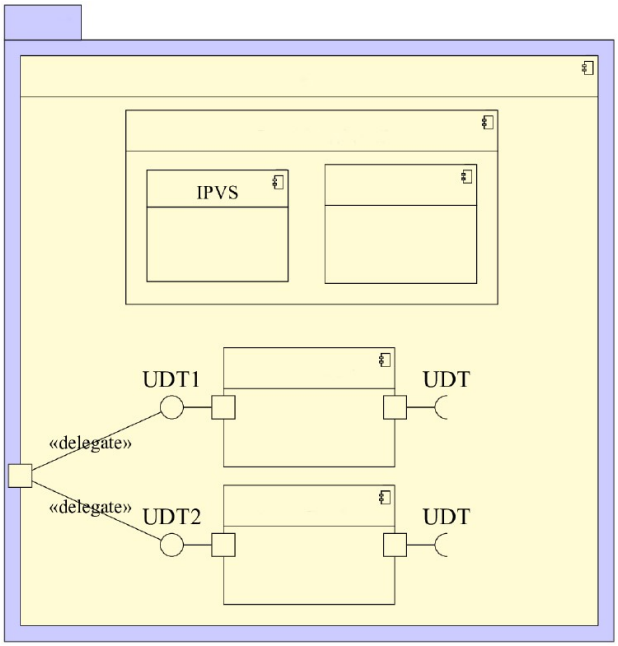

Fig. 8. Storage Deployment Diagram

A DNS server is used to update information about the location of available system elements and their availability. It acts as a key element of the information interaction of other elements of the system. To do this, each element of the system sends information about itself with a certain frequency and is authorized on the DNS server using a TCP connection (Fig. 6). In turn, other elements of the system exchange information using a data protocol based on UDT (Fig. 9). Data warehouses, as an element of a data management system, actively consume information and, accordingly, are the recipients and senders of data flows. A satellite, as an element of data consumption, also actively consumes information resources in a two-way direction. For a more detailed analysis of data flows, it is necessary to consider each element of the system in the context of data and subsystems of this element.

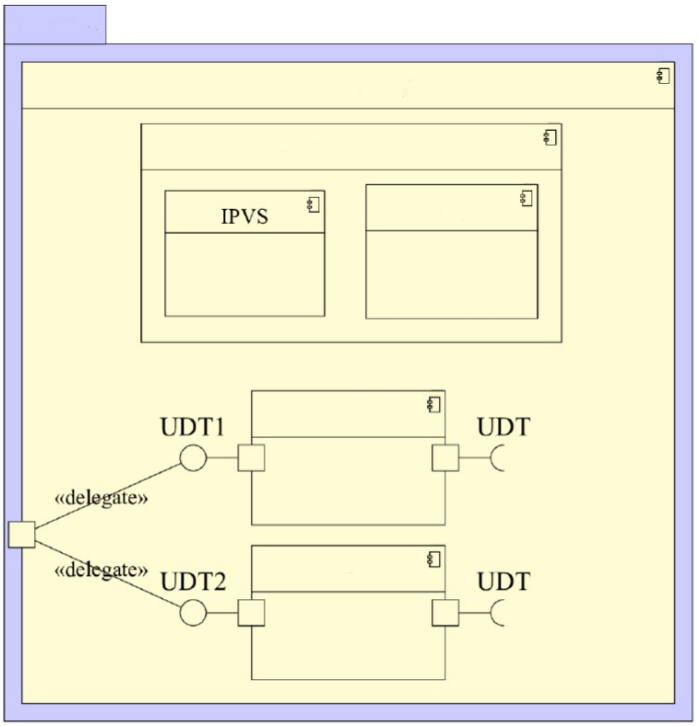

Fig. 9. Storage Units and Satellites Interaction Diagram (self-engineered)

To maximise the use of system resources, each physical server will be divided into the most independent units (containers). The containers are an operating system-level virtualization system intended for running multiple isolated instances of the Linux operating system on a single computer. They do not use virtual machines but create a virtual environment with its own process space and network stack. All the containers use a single OS kernel to maximise efficiency. To ensure the operation of a system consisting of several containers, two types are used as one system (Fig. 7, 8): a load balancer container and an application container.

The load balancer acts like a proxy server that checks the availability of containers and redirects the received requests to an available container for processing them. The application container is a full-fledged copy of the software application that processes user requests and returns the result. The data storage container (Fig. 10) receives and executes requests exclusively using the UDT protocol, for which independent UDT server and UDT client processes are running in the container. Using this protocol, the container receives data from other data storage or from clients via satellites and sends data to other data storage and clients using data access satellites.

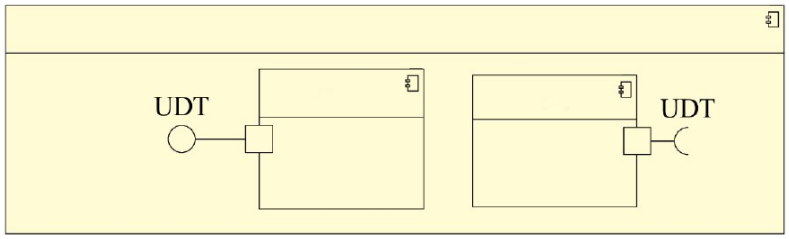

Fig. 10. Storage Container Deployment Diagram

The satellite application container (Fig. 11) contains:

1. A UDT server and UDT client for data exchange with other elements of the global data warehouse system.
2. A web server for users' easy access to data.
3. The Proftpd server is designed to expand the functionality of data access using FTP, FTPS, SFTP.

Having considered the functionality of the system as its separate business processes, one will get:

1. Search for the nearest data access satellite.
2. Uploading data to the storage.
3. Access to data.
4. Data replication.

The search for the nearest data access satellite is shown in Figure 12.

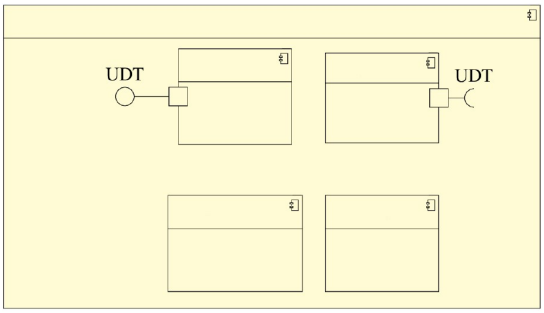

Fig. 11. Satellite Container Deployment Diagram

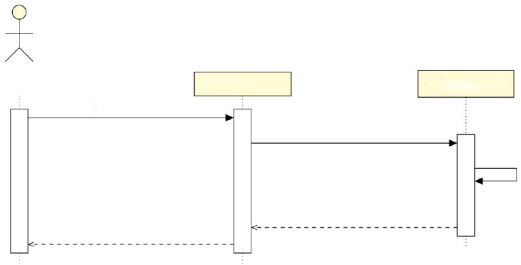

Fig. 12. Nearest Satellite Searching Sequence Diagram

When a client needs to access the data, the software application addresses the DNS client of the operating system with a request to get the IP address of the system using the domain name. In turn, the DNS client uses a network of DNS servers to access the NS servers of the system and, using the client's IP addresses, it calculates the optimal data access satellite that is "closest" (the minimum number of steps and the maximum speed) to it.

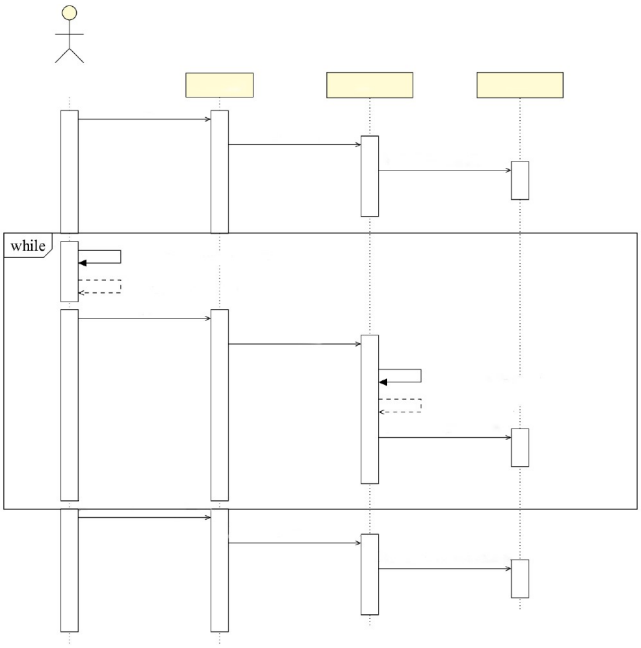

Fig. 13. Data Loading Sequence Diagram

Data is loaded to the storage according to Fig. 13. After receiving the IP addresses of the nearest satellite, the client logs in to the service and only can access the data after that. When the client needs to download a document, he or she sends the document to the satellite using TCP and a software application. The satellite, in turn, caches the data and sends them to the receiving storage directly, or through the nearest storage using UDT.

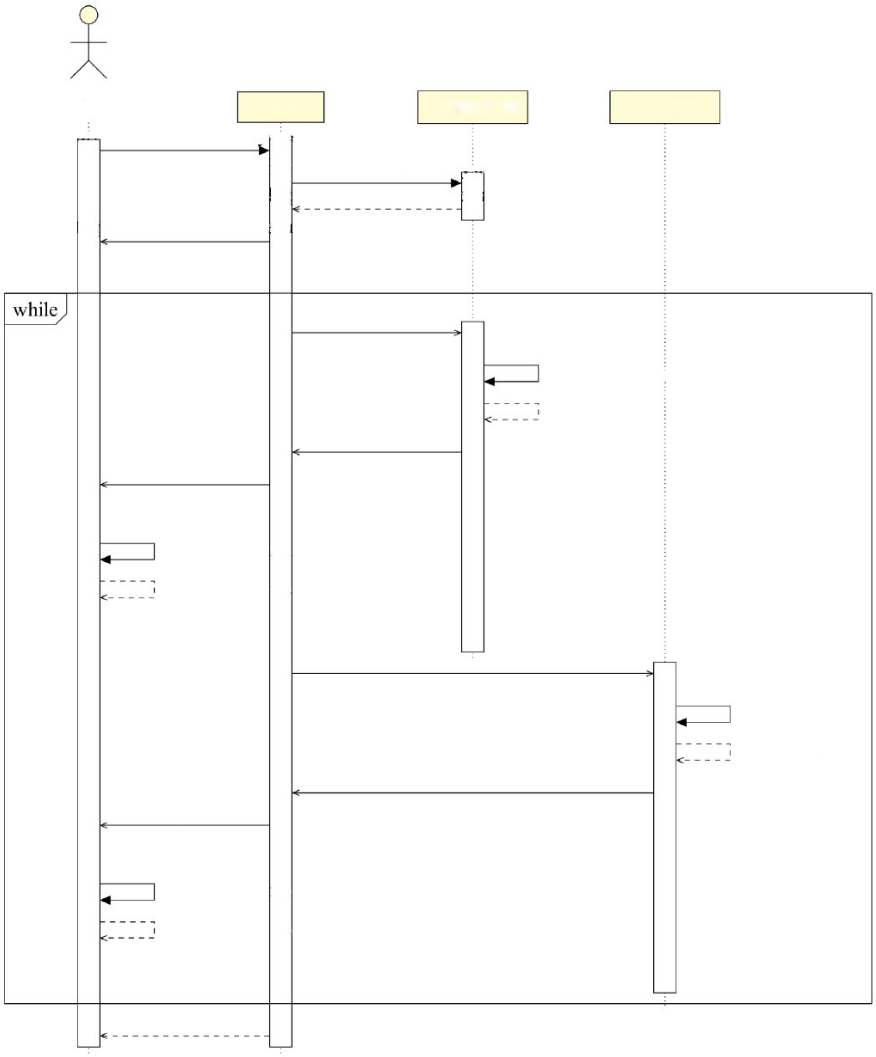

Fig. 14. Data Access Sequence Diagram

Figure 14 demonstrates data access. If it is necessary to receive the document, the user sends a request to the satellite using TCP protocol a software application to receive the document. The satellite, in turn, receives the document from the storage directly or through another storage using UDT. The document is temporarily cached on the satellite and sent to the client (TCP).

There are two approaches to data exchange in the global network: serial data transmission i.e. data transmission using one communication channel; parallel data transmission i.e. data transmission using several communication channels. Speed is the advantage of parallel data transmission. This method is used in computer peripherals for data exchange in data buses. The quality and conductivity of conductors is the main disadvantage of this approach. With different properties of the conductors, the bits in the data transmission may arrive with a delay, which leads to significant errors. In turn, serial data transmission is less dependent on conductors, since data are transmitted using one communication channel, but it is much slower compared to parallel data transmission. This approach is the most common in global networks.

Data transmission methods are also divided into synchronous data transmission; asynchronous data transmission. Asynchronous exchange is the most common form of serial communication, which involves the transmission of a data packet that contains information about the beginning and the end of data transmission, information for error control and data themselves. Since the architecture of the cloud data storage assumes both simultaneous data transmission in different directions and receiving data from different data sources, asynchronous serial data transmission was chosen. An asynchronous approach to the system design was also chosen for efficient resource allocation.

A system data transmission module can be represented as a system of states and transitions (Fig. 15). When the process is started, a data channel is created, which switches to the "waiting for events" status. Since this communication channel provides for both data transmission and data reception, this communication channel responds to the following events:

1. Input connection: during this event, the system needs to authenticate the client and add this connection to the connection pool.

2. Input data: get the data, decrypt and perform the actions provided in this data package. If there is no data for the communication channel to read, and data packets are not received, this packet is placed in the queue of data packets that are not received, and the system waits for the possibility of receiving data through this communication channel.

3. Data transmission envisages checking a connection with a remote client, forming data packets, encrypting them and transmitting them over a communication channel. If it is impossible to transmit data immediately, the data packets are moved to the packet queue and wait for the possibility of transmission (when the output communication channel is free).

4. Connection error: in this case, the system evaluates an error and makes a decision. If the system receives a false packet, or an unexpected one, the system "asks" the client to retransmit the data. If a larger system failure has occurred, such as a data encryption synchronization failure, the system breaks the connection with this client, and the client re-creates the connection and re-sends the data for which the server did not send confirmation.

Since client data is stored in the cloud data storage as a file structure, data transfer in the system is reduced to file transfer. The transfer of an individual file can be represented as the transfer of individual data packets (Fig. 16).

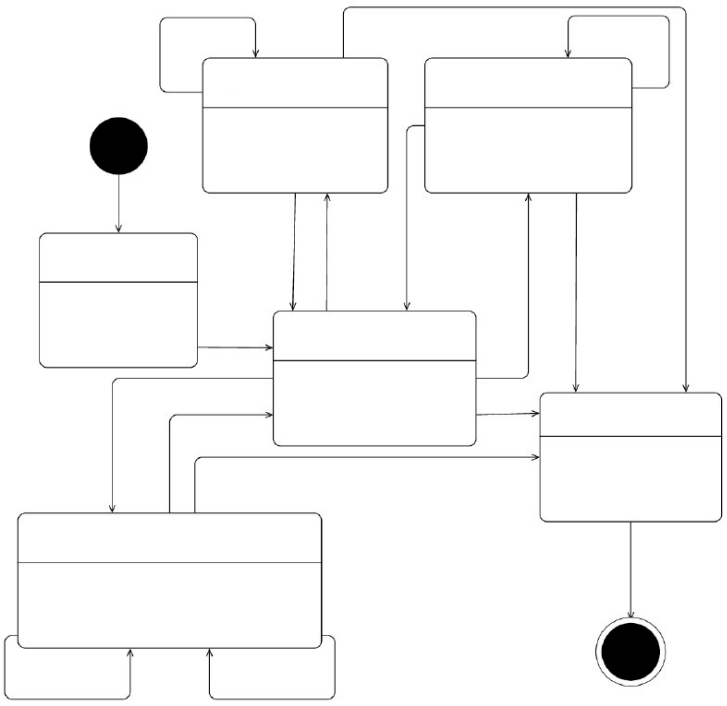

Fig. 15. Data Channel Status Diagram

During the transmission of any amount of data, the system generates data transfer packets and adds them to the data transfer queue. After that, the system performs several functions:

1. Checking for a connection to the data destination.
2. If there is no connection: creating a connection with the destination; authentication at the destination;
3. Sequential encryption and transmission of data packets (from the queue) over this communication channel.
4. If the output cache is full, the system goes into waiting for other events.

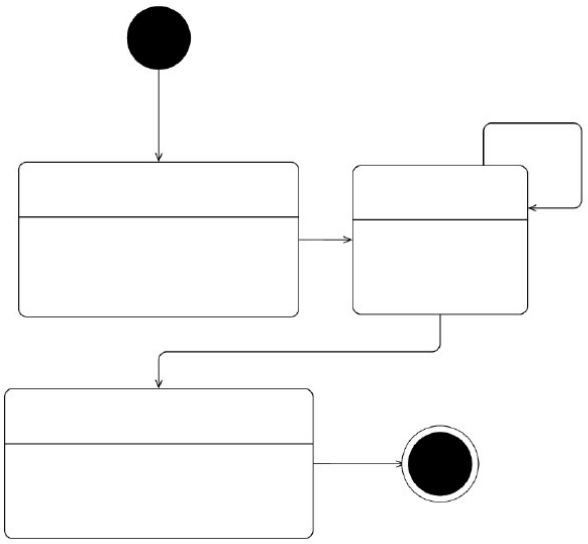

Fig. 16. If the output cache is full, the system goes into waiting for other events.

After successfully sending a data packet, the system notices the packet as sent in the data transfer queue. After the client successfully receives the data packet, it sends a confirmation packet. The server part deletes the packet from the data transfer queue after receiving this packet. In case of disconnection with the data receiver, the system automatically changes the status of the sent data packets to new ones for retransmission of data (Fig. 17). The logical model of the data transmission system can be represented in the form of a class diagram in the cloud data storage (Fig. 18). It is obvious from the class diagram that the conceptual model of the data transfer system is similar in the data warehouse and the satellite. This approach simplifies the construction of systems of this nature. It is also obvious from this structure that "Connection" is an abstract class and can be used by protocols such as TCP or UDT to transport data. This approach provides significant advantages in cases with a single limited communication channel, which is very relevant for the current state of communication.

[ ]

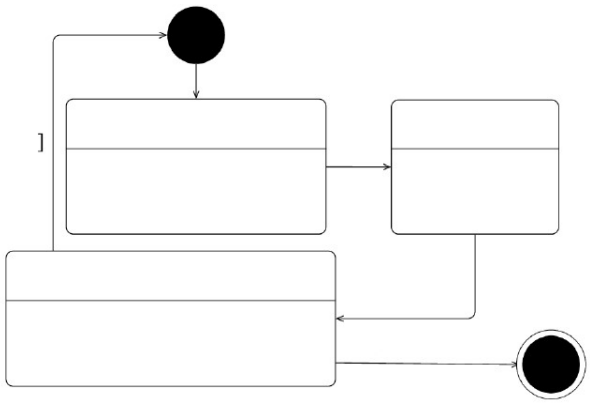

Fig. 17. Data Packet Transmission Status Diagram

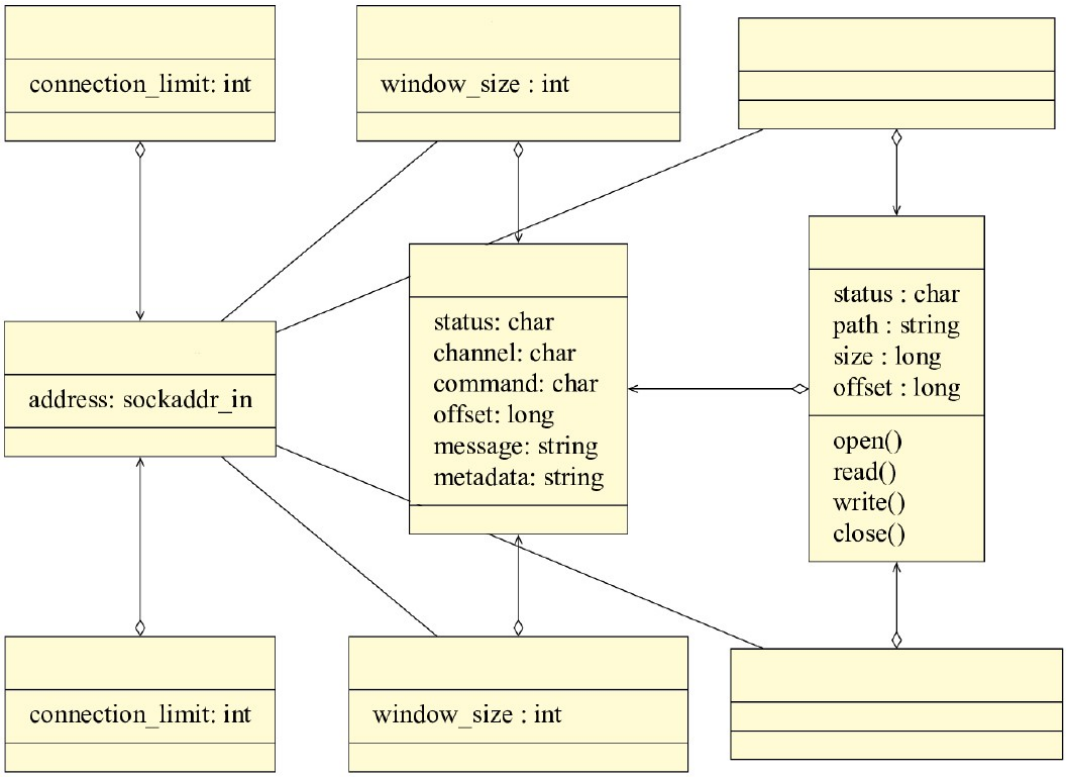

Fig. 18. System Class Diagram

If it is necessary to transfer data (a file), the system performs several simple actions:

1. Creating an instance of the "File" class.
2. Adding it to the "Source File Queue".
3. Getting a link to the connections (create it, if needed).
4. Creating instances of the "Packet" class within the free window size for the connection.
5. If the connection is ready to receive a "Packet" from the "Outgoing Packet Queue", encryption and transmission over the communication channel.
6. Receiving confirmation of the end of the file.
7. Deleting the instance of the "File" class.

On the far side of the connection, the recipient of the data (file) performs the following actions:

1. Getting a data packet with information about the data (file) and decrypting it
2. Creating an instance of "File".
3. Adding it to the "Input File Queue".
4. Creating a physical file.
5. Receiving data packets, decrypting and filling the physical file with them, sending confirmation of receiving data packets.
6. Getting a file completion package.
7. Closing the file.
8. Deleting the instance of the "File" class.
9. Sending confirmation of receipt of the full file.

The process of receiving a data packet can be divided into two stages:

– getting the packet header;
– getting the main body of the packet.

The message header contains information about:

1) The communication channel for multiplex data transmission of several files at one time;
2) The size of the data packet body;

3) The command is a type of data that is transmitted in the packet body (creating a directory, a file, data from the file, confirming receipt of the packet...); additional information may also be contained, depending on the command.

5. CONCLUSION

A session-level protocol for such networks was developed taking into account the difficulties and disadvantages of the existing state of large-volume data transmission through high-performance regionally distributed networks, which were identified during the study of the problem. The new session layer protocol is compatible with the standard OSI TCP and UDP stack protocols. In addition, it is adapted for modern networks, i.e., it has the functionality for efficient use of network bandwidth, regardless of its characteristics. When using such a protocol, the transmission delays between different types of networks do not increase. Thus, the protocol ensures efficient use of the resources of the extreme nodes of the network. In the matter of security, it supports data encryption, i.e. it can allow the connection of encryption protocols. The efficiency of the UDP-based protocol was optimised and it was shown that these ideas could be used to deploy effective and practical applications based on UDP. For example, using the UDT protocol environment (based on UDP) can easily support various congestion control algorithms, for example, high-speed TCP or explosive RBUDP.

The use of this approach made it possible to increase the performance of both the data transfer element and the system as a whole, in contrast to the methods proposed in [21; 22]. The use of multi-channel communication allowed simultaneous data transmission over one communication channel, and the universalization of the architecture allowed the use of various data exchange protocols on different parts of the network, depending on efficiency.

The authors would like to thank the management of Bunin Yelets State University for financial support of this study.

REFERENCES

1. Cong, W., Zheng, Y., Zhang, Z., Kang, Q., & Wang, X. (2017). Distributed Storage and Management Method for Topology Information of Smart Distribution Network, *Dianli Xitong Zidonghua/Automation of Electric Power Systems*, **41**(13), 111–118, doi: 10.7500/AEPS20161228008
2. Benbelgacem, S., Guezouli, L., & Seghir, R. (2020). A Distributed Information Retrieval Approach for Copyright Protection. In: *ACM International Conference Proceeding Series*. Piscataway: Institute of Electrical and Electronics Engineers, doi: 10.1145/3386723.3387882
3. Tong, B. B., Zou, G. B., & Shi, M. L. (2013). A distributed protection and control scheme for distribution network with DG, *Advanced Materials Research*, 732–733, 628–633, doi: 10.4028/www.scientific.net/AMR.732-733.628
4. Jiao, J., Yang, Y., Feng, B., Wu, S., Li, Y., et al. (2017). Distributed rateless codes with unequal error protection property for space information networks, *Entropy*, **19**(1), 38, doi: 10.3390/e19010038
5. Kornienko, D. V. (2006). On the spectrum of the Dirichlet problem for systems of operator-differential equations, *Differential Equations*, **42**(8), 1124–1133, doi: 10.1134/S0012266106080076
6. Zhu, X., Ning, Z., Fei, H., Li, Q., & Li, D. (2013). Study on protection scheme for low-voltage distribution network with distributed generation, *Information Technology Journal*, **12**(16), 3655–3659, doi: 10.3923/itj.2013.3655.3659
7. Tian, J., Gao, H., Hou, M., Liang, J., & Zhao, Y. (2010). A fast-current protection scheme for distribution network with distributed generation. In: *IET Conference Publications*. Piscataway: Institute of Electrical and Electronics Engineers, doi: 10.1049/cp.2010.0319

8. Zhong, S., Liu, C., Yang, Z., & Yan, D. (2009). Privacy protection model for distributed service system in converged network, *Proc. of International Conference on E-Business and Information System Security* (Wuhan, China), doi: 10.1109/EBISS.2009.5138026
9. Chen, X., Li, Y., Zhao, M., Wen, A., & Liu, N. (2014). A coordinated strategy of protection and control based on wide-area information for distribution network with the DG, *Proc. of International Conference on Power System Technology: Towards Green, Efficient and Smart Power System* (Chendgu, China), doi: 10.1109/POWERCON.2014.6993803
10. Song, X., Zhang, Y., Zhang, S., Song, S., Ma, J., et al. (2018). Active distribution network protection mode based on coordination of distributed and centralized protection, *Proc. of 2017 China International Electrical and Energy Conference*. (Piscataway, NJ), doi: 10.1109/CIEEC.2017.8388442
11. Kornienko, D. V. (2006). On a spectral problem for two hyperbolic systems, *Differential Equations*, **42**(1), 101–111, doi: 10.1134/S0012266106010083
12. Abbaspour, E., Fani, B., & Heydarian-Forushani, E. (2019). A bi-level multi agent-based protection scheme for distribution networks with distributed generation, *International Journal of Electrical Power and Energy Systems*, **112**, 209–220, doi: 10.1016/j.ijepes.2019.05.001
13. Kornienko, D. (2019) Spectral Properties of the Cauchy Problem for Some Linear Systems of Partial Differential Equations, *Journal of Computational and Theoretical Nanoscience*, **16**, 2780–2789, doi: 10.1166/jctn.2019.8128
14. Zhou, C., Zou, G., Yang, J., & Lu, X. (2019). Principle of Pilot Protection based on Positive Sequence Fault Component in Distribution Networks with Inverter-interfaced Distributed Generators, *Proc. of IEEE PES GTD Grand International Conference and Exposition Asia* (Bangkok, Thailand), doi: 10.1109/GTDAsia.2019.8716011
15. Maximov, R. V., Ivanov, I. I., & Sharifullin, S. R. (2017). Network topology masking in distributed information systems, *CEUR Workshop Proceedings*, **2081**, 83–87.
16. Kornienko, D. V. (2020). Organization of a system of digital education practices in the municipal sphere of general education, *Journal of Physics: Conference Series*, **1691**(1), 012108, doi: 10.1088/1742-6596/1691/1/012108
17. Kornienko, D. V., Shcherbatykh, S. V., Mishina, S. V., & Popov, S. E. (2020). The effectiveness of the pedagogical conditions for organizing the educational process using distance educational technologies at the university, *Journal of Physics: Conference Series*, **1691**(1), 012090, doi: 10.1088/1742-6596/1691/1/012090
18. Ruan, W., & Zhan, H. (2014). A new protection algorithm for distribution network with distributed generation based on intelligent electronic device information, *Lecture Notes in Electrical Engineering*, **237 LNEE**, 275–282, doi: 10.1007/978-3-319-01273-5_30.
19. Al Sukhni, E. M., & Mouftah, H. T. (2008). Availability-guaranteed distributed provisioning framework for differentiated protection services in optical mesh networks, *IEEE Globecom Workshops*, doi: 10.1109/GLOCOMW.2008.ECP.46
20. Alexandrovich, G. P. (2006). The applying sets graph protection model to the detection information threats in distributed networks and data base management systems, *Proc. of 8th International Conference on Actual Problems of Electronic Instrument Engineering* (Novosibirsk, Russia), doi: 10.1109/APEIE.2006.4292398
21. Mishina, S. V., & Kornienko, D. V. (2021). Setting up data exchange between information systems that automate accounting at the enterprise, *Journal of Physics: Conference Series*, **2094**(3), 032018, doi: 10.1088/1742-6596/2094/3/032018
22. Kornienko, D. V., Mishina, S. V., & Melnikov, M. O. (2021) The Single Page Application architecture when developing secure Web services, *Journal of Physics: Conference* Series, **2091**(1), 012065, doi: 10.1088/1742-6596/2091/1/012065

