UDC 519.688

Combinatorial problem solving method by allocating resources

Mikhail V. Khachumov, Soltan I. Salpagarov, Anton A. Mamonov, Ruslan A. Varlamov

Department of Information Technologies
Peoples' Friendship University of Russia (RUDN University)
6 Miklukho-Maklaya str., Moscow, 117198, Russian Federation

Email: khachumov-mv@rudn.ru, salpagarov-si@rudn.ru, anton.mamonov.golohvastogo@mail.ru, ravarlamov@mail.ru

There is a class of combinatorial problems that cannot be solved even on modern personal computers. You can use various means of working with high-performance computing, such as supercomputers and cloud services for working with them. However, in some situations, this approach is impractical for scientific computing due to its high cost, low scalability, and limited access.

The main goal of this work is the development and study of one of the methods for solving such problems, based on the distribution of available computing resources. For this purpose, a utility that implements this method was developed. It is a client application of a distributed computing network. Network interaction is provided by building an overlay peer-to-peer (P2P) network. P2P architecture is software implemented using UDP.

To test the resulting utility, we chose the task of forming user groups for distributing channel flows when building streaming television of the «VUD model». ViewUpload Decoupling-scheme (VUD), which strictly decouples data to what peer uploads and what it personally views. It's based on the split of downloaded user data streams into two types: the stream of the chosen TV channel, and the stream (one or more) of the other TV channel, exclusively to deliver it to other users. For the VUD model, there is a probabilistic model of data exchange between users in a homogeneous and heterogeneous in terms of the user distribution speed of a P2P network, which allows the analysis of the basic service quality indicator in streaming networks - the probability of the state of universal transmission. The calculation of this probability involves the multiple use of combinatorial formulas and, by its algorithmic complexity, belongs to the NP class problems.

As a result of the research, information was collected on the efficiency of the resource allocation method for solving combinatorial problems, the possibilities of expanding the subject area were considered, and a trial version of the distributed computing peer-to-peer network was constructed.

The main goal of this work is to develop a method for solving such problems. For this, a resource allocation model was built for two different computation algorithms. During the research the parameters of the model were refined and the current version of the program was developed. According to the results of the program test, data on the effectiveness of each of the algorithms were obtained and their comparative analysis was conducted.

Key words and phrases: combinatorial problems, complexity.

In: K. E. Samouylov, L. A. Sevastianov, D. S. Kulyabov (eds.): Selected Papers of the IX Conference "Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems", Moscow, Russia, 19-Apr-2019, published at http://ceur-ws.org

1. Introduction

As is known, despite the rapid development in the field of computers, in the modern world there are still problems with the quick solution of time-consuming tasks. Let us give an example of one of such problems [1]. When modeling a streaming television transmission system divided into sub-streams, it is necessary to use combinatorial formulas that turned out to be too voluminous for personal computers due to the high order of the number of combinations. The results of the calculations were used to construct the graph and its subsequent analysis, with the aim of selecting a set of input data for the optimal solution of the problem. Thus, it was necessary to analyze several dozen sets, despite the fact that each plotting took much longer than analysis. The question of the inefficient distribution of time arose.

The technical component for the developed method of solving time-consuming tasks was access to ten personal computers. This gave the authors the ability to build a peer-to-peer network for distributed computing. Recall that distributed computing is a way to solve time-consuming tasks, in which several computers are used, each of which solves its part in the general set of tasks. In our time, the approach to the distribution of calculations has stood out in a separate scientific area and is rapidly developing, providing a set of diverse methods and practices [2].

The purpose of this work is to develop a method for solving combinatorial problems by allocating resources.

2. Combinatorial problem formulation

It was decided to develop this method on the basis of the task of forming user groups for distributing channel flows, which is the combinatorial task of expanding balls into baskets, where the ball corresponds to the user and the basket to the distribution group of the sub-stream [1,3]. The construction of P2P television networks has recently become an increasingly common task, along with the growing number of systems using this technology [4]. Since the speed of information transmission in the network depends on external factors, the main work quality indicator of such systems is the stability of the service. To predict the operation of the system and make appropriate decisions when building P2P television networks, many stochastic indicators are calculated. [5].

The main formula for solving the selected problem is the formula for the probability of streaming video universal transmission (1).

$$PU = P(M_j \le \delta_j, j = 1, \dots, J) = \sum_{m \in M} n! \frac{p_1^{m_1}}{m_1!} \dots \frac{p_J^{m_J}}{m_J!} \tag{1}$$

For n number of viewers and J number of channels, where M_j – is a random number of viewers watching the j channel, the computational complexity of the algorithm built to calculate this formula reaches about $n * M^J$ for each set M_j, which causes difficulties in calculations even for $n, J > 100$, while real data suggest more than thousands of viewers for several hundred channels. So calculations using this formula in wxMaxima for 20 channels and 50 viewers take about ten minutes each.

In reviewing the algorithm for calculating the probability of a universal transfer [1], we can see that it assumes a complete enumeration of all possible states of the system. Therefore, as with any brute-force solution to any problem, the running time of this algorithm – is exponential. In addition, for each probable state of the system, it is necessary to carry out combinatorial calculations, the complexity of which is a factorial. Thus, the problem under consideration belongs to the class NP [6].

Since in the framework of this work we do not intend to find a polynomial solution for this problem, the main emphasis of the developed method is put on the effective distribution of the load between the available resources.

The calculation of the universal transmission probability formula [1] PU is conducted for a streaming television system with M channels and N viewers, of which N_h are

viewers with high speed u^h and N_l – with low speed u^l. The intrinsic speed of the channels is denoted by v, the required speed based on each viewer – r. As a preliminary calculation, based on Zipf's law, the estimated distribution of viewers by channels is constructed; channel popularity is recorded in the p array.

Further, for each channel among M, for any set of viewers with high and low speed, if transmission is possible, the probability of such a system state is added to the probability of the state of universal transmission.

for all m **in** M
  $PU_i = 0$
  for $x_h = \overline{0, N_h}$
    for $x_l = \overline{0, N_l}$
      if $(v + x_h u^h + x_l u^l) \geq (x_h + x_l) r)$ **then**
        $PU_i = PU_i + C_{N_h}^{x_h} p_j^{x_h} (1 - p_j)^{(N_h - x_h)} C_{N_l}^{x_l} p_j^{x_l} (1 - p_j)^{(N_l - x_l)}$

This is the basic algorithm for the developed utility, and therefore for testing the allocating resources method. Analyzing the constructed algorithm, we can note two different variants of the distribution of the conducted calculations. On the one hand, the separation can be carried out on one of the cycles, for example, the – channel cycle, on the other hand, it is possible to attract system resources by directly calculating combinatorial formulas.

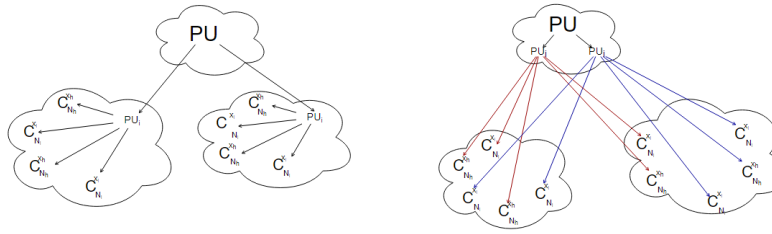

Figure 1. Distribution of computing

So, channeling involves less messaging between the participants in computing, and therefore less network load. However, with a large difference in the popularity of channels, there is a big difference in the computational complexity of the same algorithm for different channels. In addition, the number of channels must be an order of magnitude greater than the number of active nodes to ensure a uniform distribution of calculations.

With the direct separation of combinatorial formulas of factorials and combinations, the distribution of computations is much more efficient, regardless of the ratio of channels, viewers, and counting nodes. Following this mechanism of load balancing, each active node can immediately get the next part of general formula after completing the processing of the current part, since there are always obviously more such parts than nodes. However, such fragmentation of the initial task requires additional calculations, which reduces the proportion of computing power allocated to the calculation of formulas. Also, with this approach to the division of system resources, an active exchange of messages between the participants in computing is required, which can put a heavy strain on the network and cause corresponding delays.

3. Existing methods for solving time-consuming problems

The problem of time-consuming computation is not new. It is present in various scientific and domestic spheres, in each case having its own solution or not having

it at all.Many scientists are still working on optimizing of many problem solutions; supercomputers, cloud computing services, various programs and resources are involved in working with time-consuming computations.

Supercomputers have proven to be a good solution to this problem. They have a long history of exploitation, they are constantly developing and becoming more accessible [7]. Despite this, this is not an ideal solution and supercomputers have several disadvantages. By itself, the concept of a supercomputer does not allow its use by an ordinary user. Supercomputers are extremely expensive and many do not have the financial ability to rent it. And not everyone has access to such equipment since there are not so many of them and private or state-owned companies that do not allow them to use them outside of their own assignments. There is a good alternative to supercomputers that are much more accessible and hence theoretically can be used by ordinary consumers – cloud computing. But the cost of such calculations remains extremely high on the scale of a regular worker and small companies, and this option is created for medium-sized companies for which it is inexpedient to own a supercomputer

And, as any other solution for time-consuming calculation, distributed computing [8] networks have been around for a long time. There are lot of research on the parallelization problem for time-consuming global optimization [9]. The most widely known form of computing in which a virtual supercomputer is formed from clusters of loosely coupled computers – the so-called grid computing-is highlighted. The first successful projects of voluntary grid computing was bio-modelling systems, like Folding@Home [10], after which the grid technology became shortly widespread.

One of the most popular projects aimed at solving this problem is BOINC [11]. BOINC (Berkeley Open Infrastructure for Network Computing) is a grid distributed computing infrastructure based on a centralized server that coordinates volunteer computer resources. The volunteered resources can come from a variety of types of systems ranging from GPGPU (general-purpose GPU), to multiple powerful CPUs, to the ubiquitous smartphone. BOINC has been used as the underlying foundation for a number of distributed computing projects. Despite all these advantages, BOINC is not suitable for solving some tasks, as the development of the project and the attraction of volunteer capacities to it may take as much time as the solution of the task itself. Also, the system utilizes the traditional client-server model in which the consumption and supply of resources is divided and so requires server, which sometimes is inaccessible for regular user.

In this case, it is possible to pay attention to smaller projects such as the JPPF (Java Parallel Processing Framework). Simply put, JPPF enables applications with large processing power requirements to be run on any number of computers, in order to dramatically reduce their processing time. This is done by splitting an application into smaller parts that can be executed simultaneously on different machines. This system was created specifically for this reason, providing both incredible scalability and ease of use. From master/worker to P2P, with anything in between, JPPF allows any topology that will suit your requirements. Which does not require changes in a code in comparison to BOINC where throughout all the setup process it is necessary to change massive amount of code lines.

As it was mentioned in section [2], there are a number of non-regular distributed computing networks, like peer-to-peer or hierarchical ones, being developed by various research groups. Promising certain advantages, these non-regular distributed networks face new scheduling complexities. In particular, solving a set of interdependent tasks is a more complicated problem from the point of view of scheduling.

For example, the Volpex (Parallel Execution on Volatile nodes) project, is a Message Passing Interface (MPI) implementation for distributed computing networks. It joins the computing nodes to a virtual cluster suitable for applications with moderate communication requirements. Being based on MPI, this approach can address both P2P communication and possible dependence between the tasks. The implementation takes into account challenges of distributed computing networks: heterogeneity and unreliability.

4. P2P distributed computing system model

To create a utility that implements the developed method, it is necessary to first determine the current situation in which the utility will work, as well as the required functionality. As mentioned, the technical component for this was access to several personal computer numbers. With a view to the future expansion of the subject area, it was decided to generalize the calculations to the solution of various combinatorial formulas, without reference to a specific practical application. In addition, the developed method is focused primarily on the problems of numerical calculations in local networks – the subject area can serve as scientific calculations carried out in the network by universities or calculations carried out in small companies. In such tasks, the main thing is to quickly get the result of calculations for further use or analysis by the user.

To meet the task, the proposed system must comply with two fundamental postulates: generality and equality.

Former implies generalization of tasks. All requests coming to the system from the user should be generalized as much as possible and presented in the form of a set of ready-to-calculate formulas. Thus, any query is considered as a solution of a specific mathematical problem, which allows the system to quickly divide it into subtasks. When a split query is propagated to the rest of the system, only information about the required formulas, the input data, and the format of the expected output data is transmitted. The separation process can be repeated as long as it is possible and appropriate to divide complex formulas into component parts. At the same time, a preliminary analysis of the complexity of the algorithms used in the calculations allows the most optimal distribution of the load on the system.

Latter implements the concept of P2P, which has already proved itself. All network nodes must be equal to each other, having the same set of requested and provided functionality. Since all requests are provided as ready-to-calculate formulas, they are all equivalent and designed to use the same resource. Due to the analysis of the complexity of the calculations, the system can approximately calculate the execution time of various queries and, if necessary, refuse to perform too large calculations in favor of new queries. In any other case, this approach helps the system to satisfy each individual request as quickly as possible without allowing queues to accumulate.

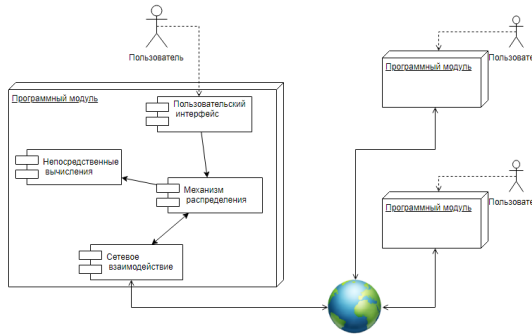

Figure 2. System model

The process of building a peer-to-peer distributed computing system is in many ways similar to that for p2p television systems. The basis of both processes is the principle of equality of participants and the distribution of the total load between nodes. With the difference in the object of distribution, where one system divides the transmission of a video stream between nodes, and the other mathematical calculations, both systems rely

on the same principles and therefore are modeled in a similar way [12]. Add to this a review of atypical grid systems [2], including peer-to-peer ones, you can get a confident representation of the model for the future system.

The immediate software model includes four main modules: user interface, allocator, calculator and network, and was built in accordance with the following model 2. The practical implementation of the target software was decided to be made in Java. Taking this into account, we have designed a UML class diagram for the future utility 3.

The user interface was built using built-in Swing library classes, such as JPanel and JButton. The main class of the program module initializes the program window, the JFrame class, which contains a panel compiling three different menus switching sets of standard buttons and forms, which ensure the user's interaction with the system.

Direct calculations are divided between a class representing a mathematical representation of a subject area and a virtual room. The Room class, which composes the Noods, passes the received requests to the Zipf class. At the same time, for the continuous provision of network communication, Room implements the Runnable programming interface, which makes it streamlined. The run method, which is a parallel stream, contains an infinite loop, listening to the network and sending the appropriate instructions when a new message is received.

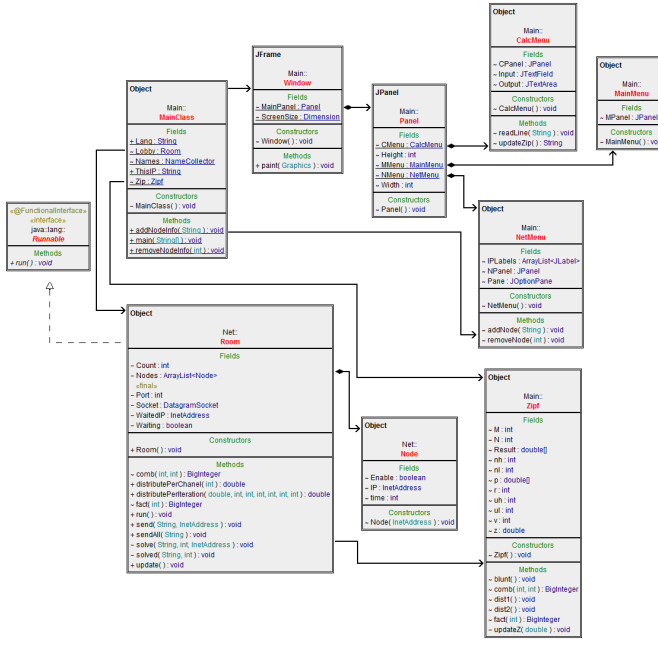

Figure 3. Class-dependence model

5. Technical aspects of the network implementation

In order to provide flexibility for the future network, it was decided to use the increasingly popular P2P direction. The concept of peer-to-peer systems, be it P2P television, P2P information exchange networks or other applications of peer-to-peer technologies, have already shown their effectiveness in providing the system with the necessary scalability, and users with the required resources [13].

While most grid computing systems, like distributed computing in general, are centralized to one degree or another, there are also a number of peer-to-peer distributed computing systems [2, 14]. There are various load-sharing methods in the P2P network [15], that provides vast opportunities to build distributed systems. This shows, that usage P2P computing, in context of distributed computing, more than acceptable.

To organize network for our utility, it is necessary to determine the required functionality from it. So, for the successful implementation of the peer-to-peer concept, each network node must simultaneously perform both server and client functions. Given the nature of the software being developed, the need is expected for a quick exchange of relatively short data packets with requests for calculations and responses. Also, there is need to create a simple connection/disconnection mechanism between system nodes.

There are two common solutions for working with the network: using the TCP [16] (Transmission Control Protocol) or UDP [17] (User Datagram Protocol) transport protocol. And while TCP, one of the most important protocols in the TCP/IP stack, provides reliable data transfer, creating TCP connections for each pair of nodes in a P2P network took an unreasonable amount of system resources, so it was decided to use UDP. The possibility of developing its own network protocol was also considered [18]. A specialized protocol usage will allow the system to significantly improve the performance of the network connection. In the future this is planned to spend more time on this problem.

The UDP protocol acts as a simple intermediary between the network layer and application services, and unlike TCP does not assume any functions to ensure the reliability of the transmission. UDP is a datagram protocol, that is, it does not establish a logical connection, does not number and does not order data packets. On the other hand, the functional simplicity of the UDP protocol determines the simplicity of its algorithm, compactness and high speed. Moreover, since the initial purpose of the developed software is work within the local network of the university, sufficient reliability of communication channels is expected without the necessity of establishing logical connections.

The specification for Internet Protocol [19] describes a minimum IP packet length of 576 bytes that all IPv4 members should support, and recommend to send larger IP packets only if you are sure that the receiving party can accept packets of this size. Therefore, to avoid fragmentation of UDP packets, as well as their possible loss, the size of the data in UDP should not exceed: (Maximum Transmission Unit) - (Max IP Header Size) - (UDP Header Size) = 1500 - 60 - 8 = 1432 bytes. In order to be sure that the packet will be received by any host, the size of the data in UDP should not exceed: (minimum IP packet length) - (Max IP Header Size) - (UDP Header Size) = 576 - 60 - 8 = 508 bytes

Thus, using the UDP protocol, you can transfer messages of substantial weight up to 508 bytes, without fear of loss of the message. Knowing that a single char character holds 2 bytes, up to 254 characters can be transmitted within a single datagram without fear of any incidental losses, which is more than enough to exchange system messages and requests.

As decided, the network interconnection of the system is based on a peer-to-peer architecture, which allows clients to easily form and operate working groups (Lobby). The network is operated using the UDP, which in this situation ensures a sufficient level of reliability and speed greater than the alternative TCP.

The network message, consisting of a virtual room and participating nodes, was mapped by two appropriate classes: Room and Node. So the Node class is a conditional client, and the Room class is a conditional server. When the software starts, a new

empty room is automatically created, where the first node immediately turns on, under the computer's own IP address. When receiving information about another active host, a new entry is created in the room, and the connected node is informed about the current state of the room. While working, receiving and sending messages, the virtual room updates information about its state including set of nodes in it, their current occupancy, the queue of requests for calculations. If any of the nodes does not contact and does not respond to the system messages sent to it, it is excluded from the room. Upon receipt of a set of requests for calculations, the room sends them to free nodes, waiting for confirmation of receipt of the request and, if necessary, repeating the sending until all requests are satisfied.

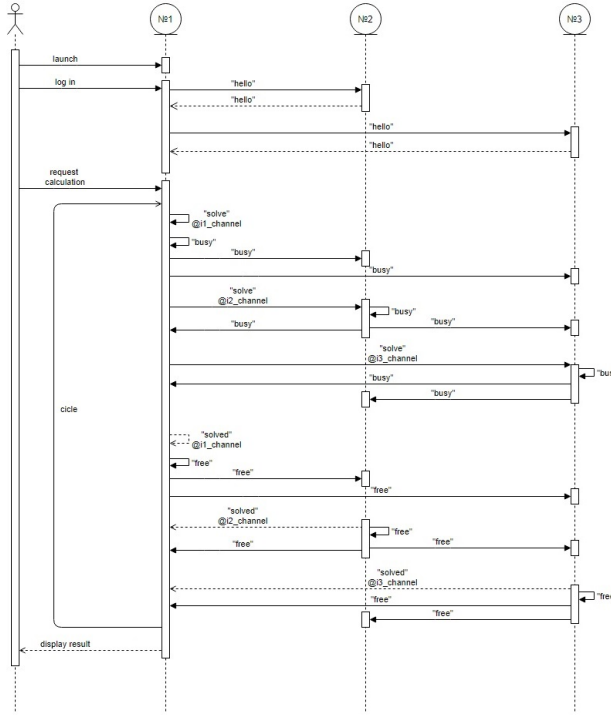

Figure 4. Runtime of calculation

The significant fields of the Node class contain the network address (the InetAddress class) and the availability boolean. The Room class aggregates the Node class (using the built-in list class ArrayList <>), contains a common constant for all applications — the port number used, and uses the DatagramSocket class to work directly with the UDP protocol.

The Node class itself has no methods, while the Room class describes a number of methods, such as solve (the method called when making a calculation request and transferring the control focus to other application modules), send (the method of sending

messages to one of the virtual room participants)), update (the method eliminating inactive nodes), and also the Runnable method of the run interface is implemented. Using this interface allows the application to listen network and conduct the necessary calculations.

To work with network messages, in our case with datagrams, Java uses the DatagramPacket class, from built-in Net library. The constructor of the DatagramPacket class receives as input a byte array defining the size of the processed packets. To receive a network message, use the receive method of the DatagramSocket class; information is retrieved using DatagramPacket's getData methods. Using the getAddress, getPort and getLength methods, you can get the sender's address, the port on which the message arrived and its length. To send a message, a new DatagramPacket instance is created with an indication of the message content, its length, the recipient's address and the active port, after which the new packet is transferred to the send method of the DatagramSocket class.

For efficient co-working nodes need to exchange some system messages to obtain information about current situation, make request and return completed answers. Such messages like:

* "hello" – is a welcome message inviting the new node into the virtual room.
* "hi" – a message automatically generated for each fixed period of time, allowing user to screen out disconnected network nodes.
* "busy" – a message sent by the node when a request is received, notifying the rest of the network about its temporary unavailability.
* "free" – a message sent by the node after the request is completed, announcing its readiness to accept the next request.
* "solve" – a request for calculations, accompanied by a formula ready for processing by the program.
* "solved" – a message on the completion of calculations, followed by a response to a request previously received by the node.
* "bye" – a message about a node leaving the network.

The image 4 shows a network message between the system nodes during the processing of a user request. When distributing calculations over channels, the node that received the request forms a queue of requests for each of the channels and sends them to the free participants of the virtual room, starting with itself. Accepting the request, the node sends a notification of temporary employment around the room, completing it returns the result and notifies the room that it is ready to accept new requests. As soon as the node that formed the request queue receives a notification about a new free neighbor, it sends the next task to it, and so on until the queue is empty.

6. Conclusion

This article has addressed the problems of time-consuming combinatorial computing. As a solution, it was proposed to develop a problem solving method by allocating resources. To achieve this goal, existing methods of working with high-performance computing, such as supercomputers and grid systems, were considered. On the basis of existing solutions and peculiarities of the P2P-TV domain, the required functionality of the future distributed computing system was determined. Having built a model and identified key points, the authors created a software module for building a peer-to-peer distributed computing system.

The module was tested on several heterogeneous nodes of various computers available for use. Combining the available computing power has greatly increased the maximum complexity of the tasks. Based on the developed method and the program module, additional studies are planned. The task of further research is to increase the efficiency of distribution algorithms, as well as expand the class of problems solved by this method [20].

Acknowledgement

The publication was prepared with the support of the "RUDN University Program 5-100".

References

1. D. Wu, Y. Liu, K. W. Ros, Queuing Network Models for Multi-Channel P2P Live Streaming Systems, IEEE INFOCOM, (2009) 73–81. doi:10.1109/INFCOM.2009.5061908
2. E. Ivashko, I. Chernov, N. Nikitina, A Survey of Desktop Grid Scheduling, IEEE Transactions on Parallel and Distributed Systems (2018) 2882–2895. doi:10.1109/TPDS.2018.2850004
3. Yu. V. Gaidamaka, E. G. Medvedeva, S. I. Salpagarov, E. V. Bobrikova, Analysis of Model for Multichannel Peer-to-Peer TV Network with View-Upload Decoupling Scheme, (2018) 123–132 doi:10.22363/2312-9735-2017-25-2-123-132
4. Y. Liu, Y. Guo, C. Liang, A survey on peer-to-peer video streaming systems, Springer Journal on Peer-to-Peer Networking and Applications, (2008) 18–28. doi:10.1007/s12083-007-0006-y
5. R. Kumar, Y. Liu, K. W. Ros, Stochastic fluid theory for P2P streaming systems, Proc. IEEE INFOCOM, (2007) 919–927 doi:10.1109/INFCOM.2007.112
6. M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to NP-Completeness, W. H. Freeman & Co. NY USA, 1990, 338 p.
7. P. E. Ceruzzi, A History of Modern Computing, MIT Press. 2003
8. A. D. Kshemkalyani, M. Singhal, Distributed Computing Principles, Algorithms, and Systems, Cambridge University Press NY USA, 2011
9. R. G. Strongin, V. P. Gergel, K. A. Barkalov, BOINC: A System for Public-Resource Computing and Storage, Lobachevskii Journal of Mathematics, (2018) doi:10.1134/S1995080218040133
10. S. M. Larson, C. D. Snow , M. R. Shirts, V. S. Pande, Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology, IEEE Transactions on Parallel and Distributed Systems (2002)
11. D. P. Anderson, BOINC: A System for Public-Resource Computing and Storage, 5th IEEE/ACM International Workshop on Grid Computing, November 8, 2004, Pittsburgh, USA. (2004) doi:10.1109/GRID.2004.14
12. A. Adamu, Yu. V. Gaidamaka, A. Samuylov, Analytical modeling of P2PTV network, Proc. 2nd International Congress on Ultra Modern Telecommunications and Control Systems (IEEE ICUMT 2010). (2010) 1115–1120 doi:10.1109/ICUMT.2010.5676520
13. J. C. Nobre, C.˙ Melchiors, C. C. Marquezan, A Survey on the Use of P2P Technology for Network Management Lobachevskii J Math, Journal of Network and Systems Management (2018) 189–221 doi:10.1007/s10922-017-9413-4
14. J. Skodzik, P. Danielis, V. Altmann, J. Rohrbeck, D. Timmermann, T. Bahls, D. Duchow, DuDE: A distributed computing system using a decentralized P2P environment, 2011 IEEE 36th Conference on Local Computer Networks, . (2011) doi:10.1109/LCN.2011.6115162
15. N. Minallah, S. Shah, N. Said, W. Khan, A. Nayab, Z. Shinwari, Performance Comparison of Chunk/Peer Scheduling Algorithms of Peer-to-Peer Streaming Systems, 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (2019) doi:10.1109/MIPR.2019.00077
16. J. Postel, INTERNET STANDARD RFC 793 Transmission Control Protocol, 1981
17. J. Postel, INTERNET STANDARD RFC 768 User Datagram Protocol, 1980
18. A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastyanov, The Method of constructing models of peer to peer protocols, 6th International Congress on

Ultra Modern Telecommunications and Control Systems and Workshops, (2014) doi:10.1109/ICUMT.2014.7002162
19. J. Postel, INTERNET STANDARD RFC 791 Internet Protocol, 1981
20. Lu. Jun, W. Tong, L. Da-xin, Research on Ordinal Properties in Combinatorics Coding Method. (2011) 51–58 doi:10.4304/jcp.6.1.51-58