Roman Cherevatenko, Artem Volokyta, Yuri Kulakov.

TRAFFIC OPTIMIZATION SYSTEM IN TRANSPORT MOBILE NETWORKS

The article considers a modified traffic balancing algorithm using software-configured networks and routing protocols: ORR and PRL. Methods of optimization which allow to reduce power consumption and to increase the maximum use of communication channels in comparison with the distributed methods are considered.

Keywords:. Traffic, traffic design, channel congestion, path load, controller, traffic balancing, graph, algorithm.

Fig.: 6. Table: 3.

Urgency of the research. Currently, with the advent of new network technologies such as software-define networks (SDN), there is a need to develop new and improve existing methods of managing network resources and build traffic.

With the increase of network size and significant power consumption problem of balancing the traffic becomes relevant. In this regard, the main objective of this project is the study and development of a method for traffic balancing in the SDN, which provides the most uniform loading of the channels of information transmission.

The analysis of features of organization and functioning of the SDN to improve the efficiency of load balancing.

Currently, with the advent of new network technologies such as software-define networks (SDN), there is a need to develop new and improve existing methods of managing network resources and build traffic.

With the increase of network size and significant power consumption problem of balancing the traffic becomes relevant. In this regard, the main objective of this project is the study and development of a method for traffic balancing in the SDN, which provides the most uniform loading of the channels of information transmission.

The analysis of features of organization and functioning of the SDN to improve the efficiency of load balancing.

A smoothing algorithm finds the corresponding new parents to switch from overloaded nodes. The routing element child of the overloaded node is instructed to switch its parent node. Such forwarding dynamic routing in General, is based on data from the local host, which easily falls into a local stimulus, while global routing algorithm improved complicates distribution and is not suitable for large-scale networks. With the advent of SDN separation of control and forwarding, and provide a more efficient way of routing algorithms in WSN. As shown in Fig. 1, a network architecture based on SDN will be more flexible and software updates easier.

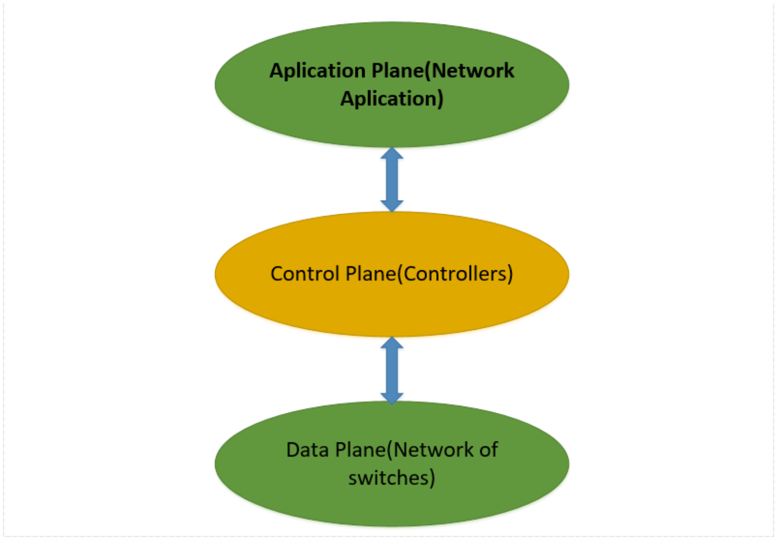

Aplication Plane(Network Aplication)
Control Plane(Controllers)
Data Plane(Network of switches)

Fig.1 General architecture of software-defining networks (SDN)

The computer networks of our time are quite large, as well as a large variety of equipment used by them. Including mobile access points. This clearly complicates the management of this type of network, as well as the design of the traffic itself.

The second main task today is to increase bandwidth and ensure the reliability of information transmission. In this regard, it is important to develop ways to organize the operation of network data centers and traffic design (TE), taking into account the features and benefits of SDN.

When considering the control device of our days (Switch or router) "Layers", contains 3 components, see Figure 2.

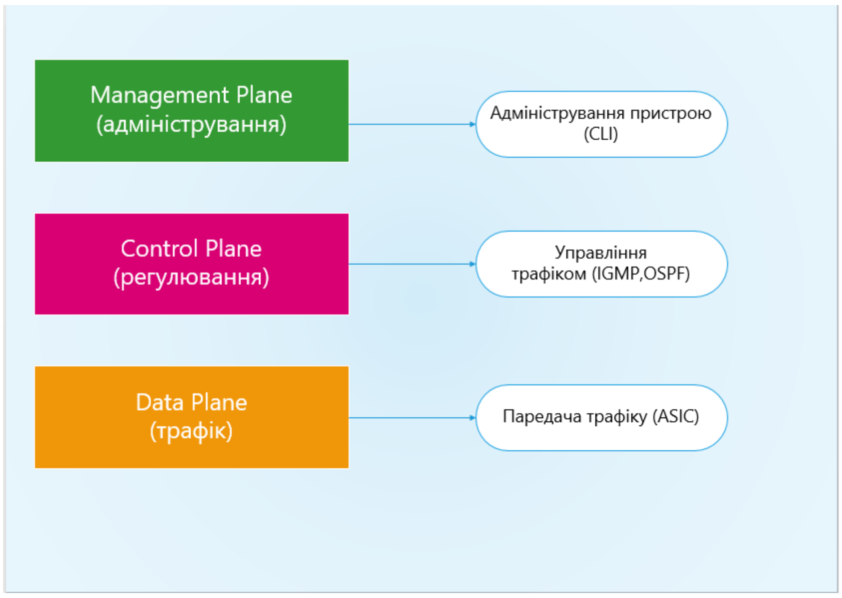

Management Plane (адміністрування)
Адміністрування пристрою (CLI)
Control Plane (регулювання)
Управління трафіком (IGMP,OSPF)
Data Plane (трафік)
Паредача трафіку (ASIC)

Fig. 2 Typical network device

According to the main principles of SDN, all control logic is in the controller, which can monitor the operation of the entire network. This is not a relatively new idea. For example, in the past, communicators used a similar logic, which they tried to implement through the modernization of telephone networks, which resulted in the emergence of "smart grids", and later Softswitch class switches.

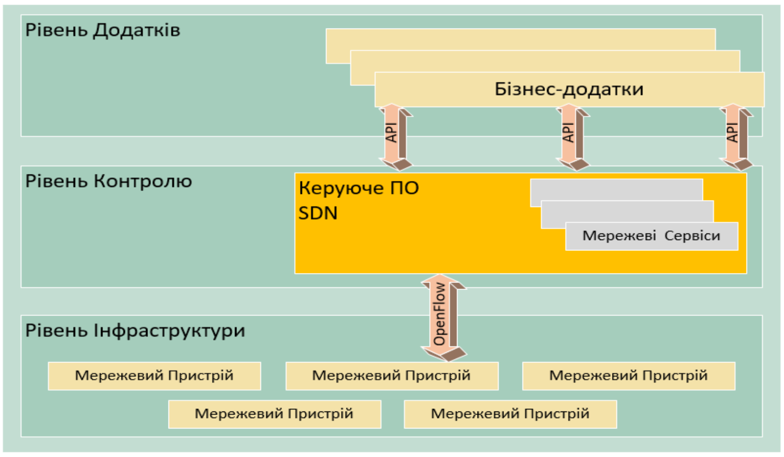


Fig. 3 Levels of operation of the data transmission algorithm

In fig. 3 shows a data transmission algorithm which contains 3 levels, namely: the level of applications, the level of control, and the level of infrastructure. Each of the layers is responsible for its part of the data transmission and contains the necessary components for this.

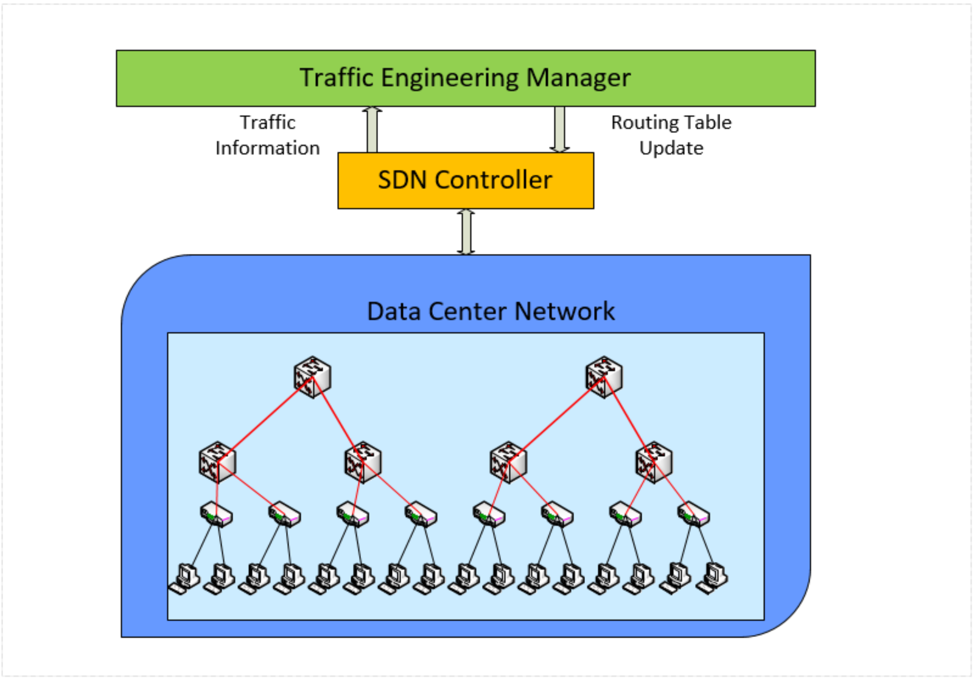


Fig. 4 General structure of the Traffic Engineering (TE) system

In fig. 4 The structure of technical traffic (TE), which includes a data center (DCN), as well as an SDN controller and a TE manager.

DCN is engaged in compiling and transmitting data to the SDN about network congestion, as well as the status of the channel. All the necessary information to design traffic to the manager of the TE provides SDN. After the controller changes the routing information in the SDN, the switching is performed by updating its own routing tables, in order to re-select the most optimal route, taking into account the minimization of power consumption and channel congestion.

Thus, we can achieve energy savings of about 40%, as well as increase the maximum channel load by 60% compared to static TE methods.

Table 1

Algorithm comparison table .

	Speed	**Fast**	**Simplicity of the algorithm**
Depth search algorithm	medium	In some cases, incorrect load results channels	Easy
Modified algorithm for finding the shortest path	Fast	Accurate channel load results	medium

Table 1 shows a comparative table with the main characteristics for comparison, namely: speed, accuracy, and simplicity of the algorithm.

In contrast to this algorithm, a modified method of traffic balancing is proposed, which takes into account the peculiarity of transport networks in SDN networks based on multi-track routing in order to increase the efficiency of computer networks.

Thus, the problem of developing a system for optimizing traffic in mobile transport networks is relevant. Which will allow you to use a more balanced load of data on information transmission channels.

Routing information is generated using a modified wave routing algorithm. This algorithm simultaneously with the formation of a given route forms the path from all intermediate nodes to the final node. This reduces the time of formation of a new track and eliminates the re-formation of route information for previously formed sections of the track. For example, for the network shown in Fig. 5, the minimum paths are presented in table 2, based on the values for which the load balancing is performed.

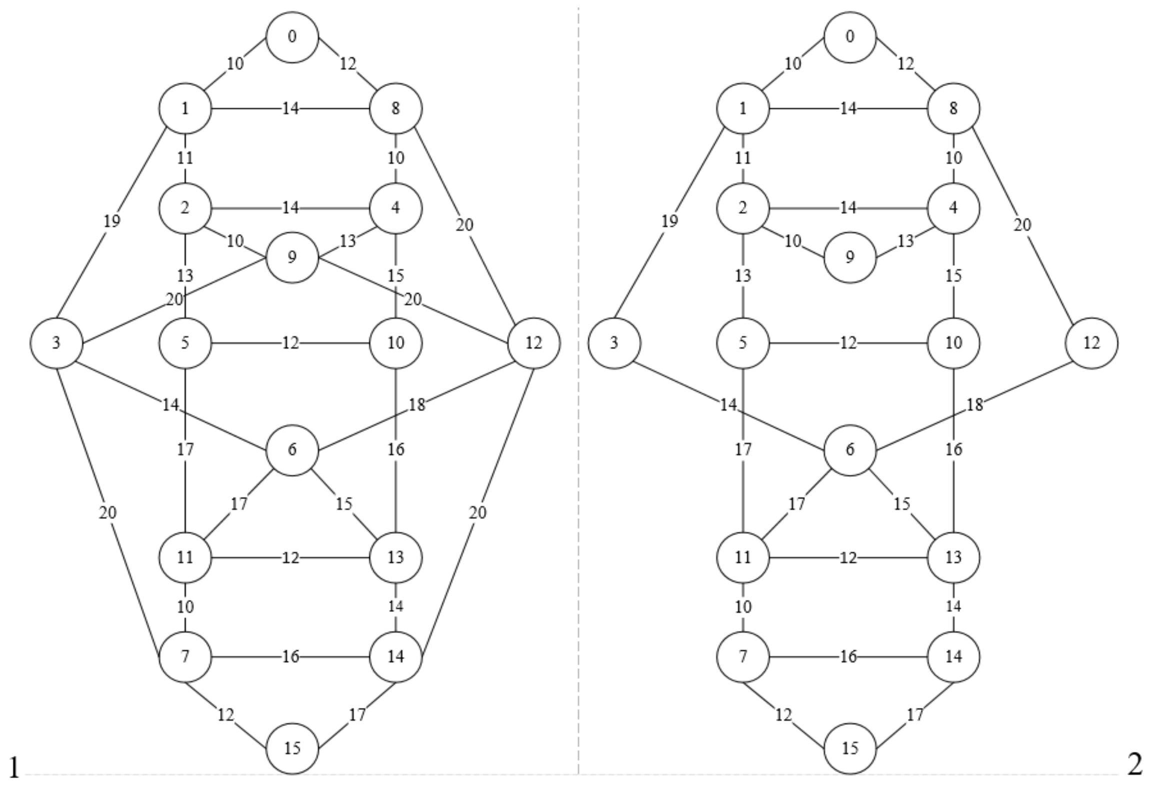


Fig. 5. Graph of data transfer between vertices

Figure 5 (1) shows the fault tolerance of distributed systems for the Hyper de Bruijn topology, the source [3] shows the use of special fail-safe topologies. Figure 5 (2) shows a lightweight Bruijn graph to reduce the degree of vertices. In Fig. 5 shows a graph of data transmission between vertices using a random initial weight of the edge. The minimum paths in (Table 2) were obtained by analyzing all possible paths from Figs.

Table 2

Table of minimum paths between the vertices of the graph

	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
R_0	0	L0.1	L0.1.2	L0.1.3	L0.8.4	L0.1.2.5	L0.8.12.6	L0.1.2.5.10.13.11.7	L0.8
R_1	L0.1	0	L1.2	L1.3	L1.2.4	L1.2.5	L1.3.6	L1.3.6.3.14.7	L1.8
R_2	L2.1.0	L2.1	0	L2.1.3	L2.4	L2.5	L2.5.10.13.6	L2.5.10.13.14.7	L2.1.8
R_3	L3.1.0	L3.1	L3.1.2	0	L3.1.2.4	L3.6.13.10.5	L3.6	L3.6.13.14.7	L3.1.8
R_4	L4.8.0	L4.2.1	L4.2	L4.2.1.3	0	L4.2.5	L4.2.5.10.13.6	L4.2.5.10.13.14.7	L4.8
R_5	L5.2.1.0	L5.2.1	L5.2	L5.10.13.6.3	L5.2.4	0	L5.10.13.6	L5.10.13.14.7	L5.2.1.8
R_6	L6.12.8.0	L6.3.1	L6.13.10.5.2	L6.3	L6.13.10.5.2.4	L6.13.10.5	0	L6.13.14.7	L6.12.8
R_7	L7.11.13.10.5.2.1.0	L7.14.13.6.3.1	L7.14.13.10.5.2	L7.14.13.6.3	L7.14.13.10.5.2.4	L7.14.13.10.5	L7.14.13.6	0	L7.14.13.6.12.8

Ended table 2

	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
R_8	L8.0	L8.1	L8.1.2	L8.1.3	L8.4	L8.1.2.5	L8.12.6	L8.12.6.13.4.7	0
R_9	L9.2.1.0	L9.2.1	L9.2	L9.4.2.5.10.13.6.3	L9.4	L9.2.5	L9.4.2.5.10.13.6	L9.4.2.5.10.13.14.7	L9.4.8
R_{10}	L10.5.2.1.0	L10.5.2.1	L10.5.2	L10.13.6.3	L10.5.2.4	L10.5	L10.13.6	L10.13.14.7	L10.5.2.1.8
R_{11}	L11.13.10.5.2.1.0	L11.6.3.1	L11.13.10.5.2	L11.6.3	L11.13.10.5.2.4	L11.13.10.5	L11.6	L11.7	L11.6.12.8
R_{12}	L12.8.0	L12.8.1	L12.8.1.2	L12.6.3	L12.8.4	L12.6.13.10.5	L12.6	L12.6.13.14.7	L12.8
R_{13}	L13.10.5.2.1.0	L13.6.3.1	L13.10.5.2	L13.6.3	L13.10.5.2.4	L13.10.5	L13.6	L13.11.7	L13.6.12.8
R_{14}	L14.13.10.5.2.1.8.0	L14.13.6.3.1	L14.13.10.5.2	L14.13.6.3	L14.13.10.5.2.4	L14.13.10.5	L14.13.6	L14.7	L14.13.6.12.8
R_{15}	L15.7.11.13.10.5.2.1.0	L15.14.13.6.3.1	L15.14.13.10.5.2	L15.14.13.6.3	L15.14.13.10.5.2.4	L15.14.13.10.5	L15.14.13.6	L15.7	L15.14.13.6.12.8

	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
R_0	L0.1.2.9	L0.1.2.5.10	L0.1.2.5.10.13.11	L0.8.12	L0.1.2.5.0.13	L0.8.1.2.5.10.13.14	L0.1.2.5.10.13.11.7.15
R_1	L1.2.9	1.2.5.10	L1.3.6.11	L1.8.12	L1.3.6.13	L1.3.6.13.14	L1.3.6.13.14.15
R_2	L2.9	L2.5.10	L2.5.10.13.11	L2.1.8. 12	L2.5.10.13	L2.5.10.13.14	L2.5.10.13.14.15
R_3	L3.6.13.10.5.2.4.9	L3.6.13.10	L3.6.11	L3.6.12	L3.6.13	L3.6.13.14	L3.6.13.14.15
R_4	L4.9	L4.2.5.10	L4.2.5.10.13.11	L4.8.12	L4.2.5.10.13	L4.2.5.10.13.14	L4.2.5.10.13.14.15
R_5	L5.2.9	L5.10	L5.10.13.11	L5.10.13.6.12	L5.10.13	L5.10.13.14	L5.10.13.14.15
R_6	L6.13.10.5.2.4.9	L6.13.10	L6.11	L6.12	L6.13	L6.13.14	L6.13.14.15
R_7	L7.14.13.10.5.2.4.9	L7.14.13.10	L7.11	L7.14.13.6.12	L7.11.13	L7.14	7.15
R_8	L8.4.9	L8.1.2.5.10	L8.12.6.11	L8.12	L8.12.6.13	L8.12.6.13.14	L8.12.6.13.14.15
R_9	0	L9.2.5.10	L9.4.2.5.10.13.11	L9.4.8.12	L9.4.2.5.10.13	L9.4.2.5.10.13.14	L9.4.2.5.10.13.14.15
R_{10}	L10.5.2.9	0	L10.13.11	L10.13.6.12	L10.13	L10.13.14	L10.13.14.15
R_{11}	L11.13.10.5.2.4.9	L11.13.10	0	L11.6.12	L11.13	L11.13.14	L11.7.15
R_{12}	L12.8.4.9	L12.6.13.10	L12.6.11	0	L12.6.13	L12.6.13.14	L12.6.13.14.15
R_{13}	L13.10.5.2.4.9	L13.10	L13.11	L13.6.12	0	L13.14	L13.14.15
R_{14}	L14.13.10.5.2.4.9	L14.13.10	L14.13.11	L14.13.6.12	L14.13	0	L14.15
R_{15}	L15.14.13.10.5.2.4.9	L15.14.13.10	L15.7.11	L15.14.13.6.12	L15.14.13	L15.14	0

Dynamic load balancing depending on the nature of the load channels, see table. 2.

Load balancing procedure. In this project, the balancing procedure is to choose from a variety of available paths a minimally loaded path with a relatively uniform loading of the channels of this path. To ensure a uniform load, it is necessary to choose a path with a minimum mean load and a minimum standard deviation of the load on the path from the mean value. To achieve this goal, it is proposed to use the criterion of loading path channels:

$$D_i = \frac{n}{N}\left(d_i^0 + \frac{1}{n}\sum_{j=1}^{n}\left(d_j - d_i^0\right)^2\right), \tag{1}$$

where: n - is the number of connections (channels) of the path P_i between the initial and final vertices V_e; N – is the number of connections of the maximum path P_m between the vertices V_s and V_s; d_i^0 – the average value of the load channels of the path P_i; d_j – Download channel $L_j \in P_i$.

$$d_i^0 = \frac{1}{n}\sum_{j=1}^{n} d_j, \tag{2}$$

where:

The ratio n / N allows us to take into account the relative path length P_i.

The value of d_i^0 – determines the less loaded path.

The standard deviation of the load of the path channels

$(\left(d_j - d_i^0\right)^2)$ characterizes the degree of uniformity of the load of the path channels.

Thus, the coefficient Di allows you to choose the most optimal way in terms of load balancing in the network.

Example of channel load balancing. As an example, consider the simulation of the load balancing process when transmitting information between the vertices W1 and W10 of the graph shown in Fig. 6 Using the above formula (1) for the average load value of the channel channels. We calculate the channel load for each of the possible paths, namely:

1. 1. W_1 W_2, W_5, W_{10}
2. 2. W_1, W_3, W_6, W_{13}, W_{10}
3. 3. W_1, W_8, W_4, W_2, W_5, W_{10}
4. 4. W_1, W_8, W_4, W_2, W_5, W_{10}…

Finding the average load of the channel path:

1.1. W_7, W_3, W_1, W_5.

1.1.1. 11 + 13 + 12 = 36

1.1.2. 36 \ 3 = 12

Next, using the average load (12), we find the deviation of the load value of the channel L from the average load of the channel.

$P\Delta = P_i - P_{cp}$

1.1.3. 12 – 11 = 1

1.1.4. 12 – 13 = 1

1.1.5. 12 – 12 = 0

At this stage, the standard deviation of the channel load is calculated.

1.1.6. 12+$((1)^2 + (1)^2)$= 14

Then the multiplication of the square deviation found above is performed by dividing the number of connections by the number of connections of the maximum path (n / N) between the vertices Vs i Vs;

1.1.7. (14*3) / 12= 3.5

Next, the load factor will be calculated for the other possible paths.

1) W_1, W_2, W_5, W_{10} = 3.5

2) W_1, W_3, W_6, $W_{13,}$ W_{10} = 10

3) W_1, W_8, W_4, W_2, $W_{5,}$ W_{10} = 9.91

4) W_1, W_8, W_4, W_9, $W_{2,}$ W_5 W_{10} = 13…

Thus, for paths from vertex 1 to vertex 10 with the lowest load factor, path 1. with D1 = 3.5. For each path presented in (Table 2), the load factors of the channel are determined, the values of which are presented in (Table 3).

Table 3

Channel load paths

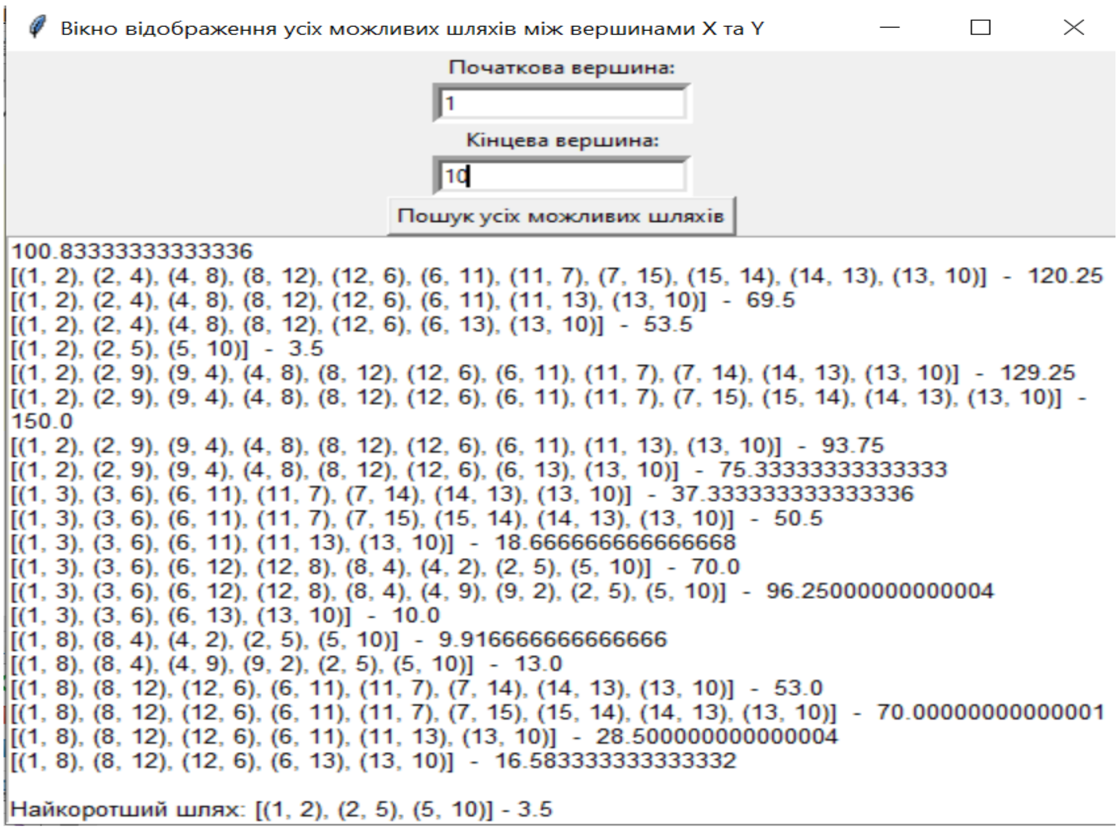
	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
R_0	0	0.71	1.83	7.85	1.85	3.42	11.8	19.9	0.85	2.35	5.07	13.4	6.4	14.0	14.0	20.2
R_1	0.71	0	0.91	1.35	2.42	2.15	4.46`	12.6	1.07	1.57	3.5	6.28	5.0	8.18	9.28	12.3
R_2	1.83	0.91	0	7.23	1.07	0.92	12.0	11.2	2.83	0.71	2.0	8.0	13.1	5.58	7.5	13.1
R_3	7.85	1.35	7.23	0	9.46	6.57	1.0	5.83	4.83	11.5	3.92	3.07	3.2	2.5	3.21	6.0
R_4	1.85	2.42	1.07	9.46	0	2.0	10.0	13.5	0.71	0.92	3.46	9.46	8.66	7.5	8.69	14.6
R_5	3.42	2.15	0.92	6.57	2.0	0	8.62	7.84	4.0	2.28	0.85	5.53	9.71	3.38	5.07	9.07
R_6	11.84	4.46	12.0	1.0	10.0	8.62	0	3.92	3.81	11.3	3.55	1.21	1.5	1.5	2.14	4.28
R_7	19.0	12.6	11.2	5.83	13.5	7.84	3.92	0	18.0	15.0	3.85	0.71	7.0	2.16	1.23	0.85
R_8	0.85	1.07	2.83	4.83	0.71	4.0	3.81	18.0	0	2.28	5.83	5.75	1.66	8.27	13.1	16.5
R_9	2.35	1.57	0.71	11.5	0.92	2.28	11.3	15.0	2.28	0	3.76	10.5	13.4	9.5	9.85	16.5
R_{10}	5.07	3.5	2.0	3.92	7.5	0.85	3.55	3.85	5.83	3.76	0	3.14	4.84	1.14	2.42	4.35
R_{11}	13.46	6.28	8.0	3.07	9.46	5.53	1.21	0.71	5.75	10.5	3.14	0	2.4	1.09	2.5	1.85
R_{12}	6.4	5.0	13.1	3.2	8.66	9.71	1.5	7.0	1.66	13.4	4.84	2.4	0	3.5	4.86	6.93
R_{13}	14.0	8.18	5.58	2.5	7.5	3.38	1.5	2.16	8.27	9.5	1.14	1.09	3.5	0	1.0	2.85
R_{14}	14.0	9.28	7.5	3.21	8.69	5.07	2.14	1.23	13.1	9.85	2.42	2.5	4.86	1.0	0	1.21
R_{15}	20.26	12.35	13.1	6.0	14.6	9.07	4.28	0.85	16.5	16.5	4.35	1.85	6.93	2.85	1.21	0

Example of comparing the load of channels after additional load. The load factors in (Table 2) were obtained by calculating the shortest path by formula (1).

Channel download path settings. We simulate a situation where the data flow between nodes loads the data channel at (0.8) and will follow the path W1, W2, W5, W10. Therefore, the load on each transition will increase by (0.8.)

In this case, the load on channels that have adjacent data paths will also increase. For example, when forming a path (L2.1.8). Path L1.8 is not changed and its weight is 14. But path L2.1 has already been used for data transmission. Therefore, it weighs 0.8 more, namely L3.7 = 11.8. Given these changes, the weight of the path L2.1.8. will change.

In this case, you need to configure all other necessary routes. As a result, the load on individual channels will increase. After clicking the "Search for all possible paths" button, the algorithm will start the calculation and the load factor of the channels of all possible paths will be displayed in the corresponding field. The re sult of the algorithm is shown in Fig. 6.


Вікно відображення усіх можливих шляхів між вершинами X та Y
Початкова вершина:
1
Кінцева вершина:
10
Пошук усіх можливих шляхів

```
100.83333333333336
[(1, 2), (2, 4), (4, 8), (8, 12), (12, 6), (6, 11), (11, 7), (7, 15), (15, 14), (14, 13), (13, 10)]  -  120.25
[(1, 2), (2, 4), (4, 8), (8, 12), (12, 6), (6, 11), (11, 13), (13, 10)]  -  69.5
[(1, 2), (2, 4), (4, 8), (8, 12), (12, 6), (6, 13), (13, 10)]  -  53.5
[(1, 2), (2, 5), (5, 10)]  -  3.5
[(1, 2), (2, 9), (9, 4), (4, 8), (8, 12), (12, 6), (6, 11), (11, 7), (7, 14), (14, 13), (13, 10)]  -  129.25
[(1, 2), (2, 9), (9, 4), (4, 8), (8, 12), (12, 6), (6, 11), (11, 7), (7, 15), (15, 14), (14, 13), (13, 10)]  -
150.0
[(1, 2), (2, 9), (9, 4), (4, 8), (8, 12), (12, 6), (6, 11), (11, 13), (13, 10)]  -  93.75
[(1, 2), (2, 9), (9, 4), (4, 8), (8, 12), (12, 6), (6, 13), (13, 10)]  -  75.33333333333333
[(1, 3), (3, 6), (6, 11), (11, 7), (7, 14), (14, 13), (13, 10)]  -  37.333333333333336
[(1, 3), (3, 6), (6, 11), (11, 7), (7, 15), (15, 14), (14, 13), (13, 10)]  -  50.5
[(1, 3), (3, 6), (6, 11), (11, 13), (13, 10)]  -  18.666666666666668
[(1, 3), (3, 6), (6, 12), (12, 8), (8, 4), (4, 2), (2, 5), (5, 10)]  -  70.0
[(1, 3), (3, 6), (6, 12), (12, 8), (8, 4), (4, 9), (9, 2), (2, 5), (5, 10)]  -  96.25000000000004
[(1, 3), (3, 6), (6, 13), (13, 10)]  -  10.0
[(1, 8), (8, 4), (4, 2), (2, 5), (5, 10)]  -  9.916666666666666
[(1, 8), (8, 4), (4, 9), (9, 2), (2, 5), (5, 10)]  -  13.0
[(1, 8), (8, 12), (12, 6), (6, 11), (11, 7), (7, 14), (14, 13), (13, 10)]  -  53.0
[(1, 8), (8, 12), (12, 6), (6, 11), (11, 7), (7, 15), (15, 14), (14, 13), (13, 10)]  -  70.00000000000001
[(1, 8), (8, 12), (12, 6), (6, 11), (11, 13), (13, 10)]  -  28.500000000000004
[(1, 8), (8, 12), (12, 6), (6, 13), (13, 10)]  -  16.583333333333332

Найкоротший шлях: [(1, 2), (2, 5), (5, 10)] - 3.5
```

Fig. 6 The result of the program

Conclusions. This project proposes a method of traffic balancing, which by taking into account the peculiarities of the SDN, in particular, due to the presence of a central controller in the network, reduces the time required to form multiple routes and simplify the process of traffic balancing. Due to the many routes it is possible to virtually eliminate the delay or loss of packets in the case of traffic re-routing. In particular, the more paths generated in the SDN controller, the less likely it is that packets will be delayed or lost.

The proposed method of generating route information makes it possible to provide a more uniform loading of information transmission channels at given QoS parameters.

Further improvement of traffic balancing methods is associated with the use of methods of decision theory in the choice of path in order to predict the nature of changes in the load in the communication channels in a timely manner data transmission. This will reduce the frequency of path reconfiguration and provide a more even load on the network.

References

1. Kulakov, Y., Kogan, A.: The method of plurality generation of disjoint paths using horizontal exclusive scheduling. Adv. Sci. J. 10, 16–18 (2014). https://doi.org/10.15550/ASJ.2014.10. ISSN 2219-746X

2. Rajasekaran K., Balasubramanian K.: Energy Conscious based Multipath Routing Algorithm in WSN. Int. J. Comput. Netw. Inf. Secur. (IJCNIS), 1, 27-34 (2016). https://doi.org/10.5815/ijcnis.2016.01.04 in MECS (http://www.mecs-press.org/)

3. H. Loutskii, A. Volokyta, P. Rehida, O. Honcharenko, B. Ivanishchev and A. Kaplunov, "Increasing the fault tolerance of distributed systems for the Hyper de Bruijn topology with excess code," 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 1-6, doi: 10.1109/ATIT49449. 2019.9030487.

4. Han, Y., Seoy, S., Li, J., Yooy, J., H., Hong, J., W. : Software Defined Networking-based Traffic Engineering for Data Center Networks: In Asia-Pacific Network Operations and Management Symposium (2014), https://ieeexplore.ieee.org/document/6996601

5. Kumar, P., Dutta, R., Dagdi, R., Sooda, K., Naik, A.: A programmable and Managed Software Defined Network. Int. J. Comput. Netw. Inf. Secur. (IJCNIS) 12, 11–17 (2017). https://doi.org/10.5815/ijcnis.2017.12.02. In MECS http://www.mecs-press.org/. Accessed Dec 2017.

6. Yurii Kulakov Traffic Orchestration in Data Center Network based on Software-Defined Networking Technology Yurii Kulakov, Alla Kohan, Sergii Kopychko International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2019: Advances in Computer Science for Engineering and Education II pp 228-237

AUTORS

Roman Cherevatenko – student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

E-mail: r.cherevatenko1@gmail.com.

Artem Volokyta (supervisor) – associate professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Yurii Kulakov (supervisor) – professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.