

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ  
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»  
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ  
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»  
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова  
праця на правах рукопису

ЧЕРЕВАТЕНКО ОЛЕКСІЙ ВОЛОДИМИРОВИЧ

УДК 004.77

## ДИСЕРТАЦІЯ

СПОСІБ ПОБУДОВИ ВІРТУАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ НА  
ОСНОВІ ТЕХНОЛОГІЇ SDN

123 Комп'ютерна інженерія  
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей,  
результатів і текстів інших авторів мають посилання на відповідне джерело

---

Науковий керівник: Кулаков Юрій Олексійович, д. т. н., професор

Київ – 2025

## АНОТАЦІЯ

Череватенко О.В. Спосіб побудови віртуальної комп'ютерної мережі на основі технології SDN. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп'ютерна інженерія з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2025.

Дисертаційна робота присвячена побудові віртуальної комп'ютерної мережі для передачі заданого типу трафіку з використанням технології програмно-визначених мереж (*software-defined networks, SDN*). По результатам дослідження створено комплексний спосіб побудови віртуальної програмно-визначеної мережі, яка використовує метод побудови каналів зв'язку на основі інтегрального критерію з урахуванням метрик якості обслуговування та метод динамічної реконфігурації мережі з використанням відмовостійких резервних каналів зв'язку.

Метою дисертаційної роботи є розробка комплексного способу побудови комп'ютерної мережі для підвищення швидкості та надійності передачі даних в ній, що дозволить гарантувати відповідність мережі заданим параметрам якості обслуговування під час виходу з ладу окремих її ділянок шляхом інтеграції технологій віртуальних мереж та програмно-визначених мереж.

Досліджені відомі рішення побудови віртуальних комп'ютерних мереж з використанням технології SDN. На основі аналізу літературних джерел зроблено постановку задачі про необхідність розробки рішення для побудови віртуальної комп'ютерної мережі на основі технології SDN, яке дозволить вдосконалити процеси передачі даних у випадку відмов окремих ділянок комп'ютерної мережі. Після проведеного аналізу літератури на тему відповідності рішень побудови програмно-визначених мереж критеріям якості обслуговування (*quality of service, QoS*) зроблено висновок про необхідність розробки методу врахування якомога більшої (не менше чотирьох, оскільки відомі рішення зазвичай враховують дві або три) кількості метрик якості обслуговування при передачі даних у мережі.

Вперше розроблено та обґрунтовано інтегральний критерій побудови каналів

зв'язку у віртуальній комп'ютерній мережі на основі технології *SDN*, який, на відміну від існуючих способів побудови, одночасно враховує необхідну кількість та пріоритети метрик якості обслуговування каналів зв'язку мережі, що дозволяє досягти збереження стабільної швидкості передачі трафіку будь-якого типу. Цей критерій формується на основі числових метрик якості обслуговування зв'язків між комутаторами для оптимізації побудови шляхів під час маршрутизації.

Вперше розроблено та обґрунтовано критерій надійності вузлів мережі для побудови каналів зв'язку, який, на відміну від існуючих способів побудови, враховує здатність проміжних пристроїв мережі транспортувати трафік між своїми портами, що дозволяє збільшити надійність каналів зв'язку та зменшити втрати даних під час передачі. Використання метрики надійності вузлів у якості критерію побудови дозволяє обирати такі канали зв'язку, що задовольняють необхідній якості обслуговування при передачі різних типів трафіку.

Вперше розроблено та обґрунтовано метод побудови відмовостійких каналів зв'язку віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на розділенні віртуальної мережі на підмережі та використовує розроблені інтегральний критерій побудови каналів зв'язку з урахуванням якості обслуговування та критерій надійності вузлів, що дозволяє забезпечити стабільну швидкість, зменшити затримку та втрати даних під час передачі будь-якого типу трафіку у разі виходу з ладу окремих ділянок мережі. Використання пріоритезації метрик при побудові шляхів передачі дозволяє збільшити гнучкість застосування методу та змінювати конфігурацію віртуальної мережі в залежності від різних типів трафіку, що передається.

Удосконалено метод динамічної реконфігурації віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на використанні відмовостійких резервних каналів зв'язку, збережених у пам'яті контролера *SDN*, що дозволяє підтримувати безперервний сеанс передачі трафіку у разі відмови основного каналу зв'язку. Наявність резервного каналу дозволяє не переривати з'єднання між вузлами у разі відмови основного маршруту, поки контролер виконує побудову нового шляху в обхід проблемних ділянок та завантаження його

в комутатори.

Розроблено програмне забезпечення для *SDN*-контролера *Open Network Operating System (ONOS)*, зокрема застосунок, який реалізує розроблені в роботі методи побудови каналів зв'язку на основі інтегрального критерію та динамічної реконфігурації мережі з використанням резервних каналів зв'язку. Експерименти з використанням застосунку показали зменшення часу реконфігурації мережевої моделі та доставки даних, порівняно зі стандартними засобами маршрутизації у *ONOS*. Це було досягнуто не лише за рахунок використання інтегрального критерію побудови каналів зв'язку, а й ефективнішої компіляції інструкцій контролера та перебудові шляхів між комутаторами на основі зібраних даних про потоки. Використання технології *SDN* дозволяє виконувати переналаштування мережі на рівні контролера у разі відмови певних вузлів, а програмне забезпечення *ONOS* дозволяє вносити зміни та розширювати набір інструкцій для віртуальної мережі без зупинки контролера та процесу передачі трафіку.

Виконано побудову віртуальних мереж різної топологічної організації, що використовують запропоновані в роботі методи побудови каналів зв'язку на основі інтегрального критерію та динамічної реконфігурації на основі резервних каналів зв'язку. Створені моделі віртуальної мережі у програмній утиліті *mininet*, виконано підключення до неї централізованого *SDN* контролера зі встановленим на ньому програмним забезпеченням *ONOS*, задіяння застосунку для передачі даних між вузлами мережі. У ході експерименту генерувався трафік різних типів і відбувалося зняття показників швидкості передачі даних та їхнього об'єму при поступовому відключенні вузлів мережі. Після відключень вузлів контролер відслідковував зміни в конфігурації мережі та виконував побудову та вибір оновлених каналів передачі трафіку. За результатами експериментів, для топологічних графів різної організації вдалося передати об'єм даних, що є на 65% більшим, ніж той, що був переданий з використанням існуючих методів побудови шляхів. Вдалося не допустити зниження швидкості передачі більше ніж на 41% після відключення половини вузлів мережі.

Розроблений спосіб побудови віртуальної мережі на основі технології *SDN*

дозволив розробити експериментальну модель для практичного використання та впровадження у корпоративних *SDN* мережах з відповідним апаратним забезпеченням.

*Ключові слова:* SDN, багатошляхова маршрутизація, віртуальна мережа, інформаційна система, якість обслуговування, відомі маршрути, відмовостійкість, топологія.

## ABSTRACT

*Cherevatenko O.V.* Method for creating a virtual computer network based on SDN technology. – Qualified scientific work as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 – Computer engineering in the field of knowledge 12 – Information technology. – National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2025.

The dissertation is devoted to the construction of a virtual computer network for the transmission of a given type of traffic using software-defined networks (SDN) technology. Based on the results of the research, a comprehensive method for constructing a virtual software-defined network has been created, which uses the method of constructing communication channels based on an integral criterion considering the metrics of their quality of service and the method of dynamic network reconfiguration using fault-tolerant backup data transmission channels.

The purpose of the dissertation is to develop a comprehensive method for constructing a computer network to increase the speed and reliability of data transmission in it, which will ensure that the network meets the specified quality of service parameters during the failure of its individual sections by integrating virtual network technologies and software-defined networks.

Known solutions for constructing virtual computer networks using SDN technology have been studied. Based on the analysis of literature sources, the problem of the need to develop a solution for building a virtual computer network based on SDN technology, which will allow improving data transmission processes in the event of failures of individual sections of the computer network, was formulated. After analyzing the literature on the compliance of solutions for building software-defined networks with quality of service (QoS) criteria, the conclusion was made about the need to develop a method for considering as many (at least four, since known solutions usually consider two or three) quality of service metrics as possible when transmitting data in the network.

For the first time, an integral QoS criterion was developed and substantiated for constructing communication channels in a virtual computer network based on SDN

technology, which, unlike existing construction criteria, simultaneously considers an unlimited number of QoS metrics for network communication channels and the priorities of these metrics, which allows maintaining the data transfer rate for any type of traffic. This criterion is formed on the basis of numerical quality of service metrics for connections between switches to optimize path construction during the routing process.

For the first time, a reliability criterion for network nodes has been developed and substantiated for constructing communication channels, which, unlike existing construction criteria, considers the ability of intermediate network devices to transport traffic between their ports, which allows increasing the reliability of traffic transmission over communication channels and reducing data loss. Using the reliability metric of nodes as a construction criterion allows choosing such communication channels that satisfy the required quality of service when transmitting different types of traffic.

For the first time, a method for constructing communication channels of a virtual network based on SDN technology has been developed and substantiated, which, unlike existing construction methods, uses an integral criterion for the quality of service of network communication channels, a criterion for the reliability of network nodes and its division into subnetworks, which allows maintaining speed, reducing delay and data loss during their transmission for any type of traffic in the event of failure of individual network sections. The use of metric prioritization when constructing transmission paths allows increasing the flexibility of the method and changing the configuration of the virtual network depending on different types of transmitted traffic.

The method of dynamic reconfiguration of a virtual SDN network has been improved, which, unlike existing methods, is based on the use of fault-tolerant backup data transmission channels stored in the memory of the SDN controller, which allows maintaining a traffic transmission session in the event of a failure of the main communication channel. The presence of a backup path allows not to interrupt the connection between nodes in the event of a failure of the main route, while the controller is building a new path bypassing problem area and loading it into the switches.

Software for the Open Network Operating System (ONOS) SDN controller has been developed, in particular an application that implements the methods developed in the

work for constructing communication channels based on the integral criterion and dynamic network reconfiguration using redundant communication channels. Experiments using the application have shown a reduction in the time for reconfiguring the network model and delivering data, compared to standard routing tools in ONOS. This was achieved not only by using the integral criterion for constructing communication channels, but also by more efficient compilation of controller instructions and rebuilding paths between switches based on collected data about flows. The use of SDN technology allows for network reconfiguration at the controller level in the event of failure of certain nodes, and the ONOS software allows for changes and expansion of the set of instructions for the virtual network without stopping the controller and the traffic transfer process.

Virtual networks of different topological organizations were constructed, using the methods proposed in the work for constructing communication channels based on the integral criterion and dynamic reconfiguration based on backup data transmission channels. Virtual network models were created in the mininet software utility, a centralized SDN controller with ONOS software installed on it was connected to it, and an application was launched for data transmission between network nodes. During the experiment, traffic of different types was generated and data transmission speed and volume were measured during the gradual disconnection of network nodes. After the nodes were disconnected, the controller monitored changes in the network configuration and constructed and selected updated traffic transmission channels. According to the test results, for topological graphs of different organizations, it was possible to transmit a data volume that is 65% larger than that transmitted using existing methods for constructing paths. It was possible to prevent a decrease in transmission speed by more than 41% after disconnecting half of the network nodes.

The developed method of building a virtual network based on SDN technology allowed the development an experimental model for practical use and implementation in corporate SDN networks with appropriate hardware.

*Keywords:* SDN, multipath routing, virtual network, information system, quality of service, known routes, fault tolerance, topology.



## СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

*Наукові праці, в яких опубліковано основні наукові результати дисертації:*

1. D. Korenko, O. Cherevatenko, V. Rusinov, and Y. Kulakov, “Creation of the method of multipath routing using known paths in software-defined networks,” *Technology audit and production reserves*, vol. 4, no. 2(66), pp. 19–24, Aug. 2022, doi: <https://doi.org/10.15587/2706-5448.2022.262787>.
2. Ю. Кулаков та О. Череватенко, «Моделювання маршрутизації з використанням відомих маршрутів за допомогою SDN-контролера ONOS,» *Проблеми інформатизації та управління*, 2, № 74, сс. 55–61, 2023, doi: <https://doi.org/10.18372/2073-4751.74.17882>.
3. A. Volokyta et al., “Fault Tolerance Exploration and SDN Implementation for de Bruijn Topology based on betweenness Coefficient,” *International journal of computer network and information security*, vol. 16, no. 1, pp. 97–112, Feb. 2024, doi: <https://doi.org/10.5815/ijcnis.2024.01.08>.
4. O. Cherevatenko and Y. Kulakov, “Method of Increasing Data Transmission Stability in Software Defined Network Considering Metrics of Quality of Service,” *Information, Computing and Intelligent systems*, no. 4, pp. 37–47, 2024, doi: <https://doi.org/10.20535/2786-8729.4.2024.303467>.
5. A. Volokyta, A. Kogan, O. Cherevatenko, D. Korenko, D. Oboznyi, and Y. Kulakov, “Traffic Engineering with Specified Quality of Service Parameters in Software-defined Networks,” *International Journal of Computer Network and Information Security*, vol. 16, no. 5, pp. 1–13, Oct. 2024, doi: <https://doi.org/10.5815/ijcnis.2024.05.01>.

*Апробація наукових результатів дисертації:*

6. Ю. Кулаков та О. Череватенко, «Динамічна реконфігурація комп'ютерних мереж на основі технології SDN,» *Збірник тез доповідей XV Міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології», 25–26 квітня 2024 р.,* сс. 104–105, 2024. [Online] Доступ: <https://csnt.nau.edu.ua/files/2024/sbirnyk2024.pdf>
7. O. Cherevatenko, Y. Kulakov, “Increasing the data transmission speed considering QoS parameters in SDN networks under the ONOS controller management.” *The International Conference on Security, Fault Tolerance, Intelligence*, Kyiv, Ukraine, June 28, 2024. [Online] Available at: <https://icsfti-proc.kpi.ua/issue/view/18065>.

*Праці, які додатково відображають результати дисертації:*

8. О. Гончаренко та О. Череватенко, «Способи мультиканальної маршрутизації в мережах надлишкового Де Бруйна,» *Технічні науки і технології*, № 2(24), сс. 123–130, 2021, doi: [https://doi.org/10.25140/2411-5363-2021-2\(24\)-123-130](https://doi.org/10.25140/2411-5363-2021-2(24)-123-130).

## ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ .....	13
ВСТУП .....	14
РОЗДІЛ 1. АНАЛІЗ ВІДОМИХ РІШЕНЬ ПОБУДОВИ ПРОГРАМНО- ВИЗНАЧЕНИХ МЕРЕЖ .....	20
1.1. Проблема зростання об'єму трафіку у комп'ютерних мережах .....	20
1.2. Концептуальний аналіз технології віртуальних мереж .....	25
1.3. Аналіз архітектурних особливостей мереж SDN .....	28
1.4. Аналіз відомих наукових підходів до побудови мереж SDN .....	33
1.5. Аналіз відомих наукових підходів до підвищення відповідності SDN мереж якості обслуговування .....	38
Висновки до розділу 1 .....	45
РОЗДІЛ 2. ПОБУДОВА КАНАЛІВ ЗВ'ЯЗКУ ВІРТУАЛЬНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN ІЗ ЗАДАНОЮ ЯКІСТЮ ОБСЛУГОВУВАННЯ.....	47
2.1. Математичне обґрунтування проблеми побудови каналів зв'язку з урахуванням параметрів якості обслуговування .....	47
2.2. Метод побудови каналів зв'язку на основі інтегрального критерію з урахуванням метрик якості обслуговування .....	50
2.2.1. Обчислення інтегрального критерію побудови каналу зв'язку.....	50
2.2.2. Математичне обґрунтування зменшення часової складності побудови віртуальних каналів зв'язку .....	54
2.2.3. Обчислення критерію надійності вузлів для зменшення втрат нееластичного трафіку у мережі SDN .....	62
Висновки до розділу 2 .....	67
РОЗДІЛ 3. УДОСКОНАЛЕННЯ МЕТОДУ ДИНАМІЧНОЇ РЕКОНФІГУРАЦІЇ ВІРТУАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN ...	69
3.1. Реалізація методу побудови каналів зв'язку на основі інтегрального критерію з урахуванням якості обслуговування .....	69
3.2. Пріоритезація каналів передачі даних на основі критерію надійності вузлів	

мережі.....	78
3.3. Метод динамічної реконфігурації віртуальної мережі на основі технології SDN з використанням резервних каналів зв'язку .....	83
Висновки до розділу 3 .....	87
РОЗДІЛ 4. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОБУДОВИ ВІРТУАЛЬНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN.....	88
4.1. Аналіз технічних можливостей протоколу OpenFlow для побудови віртуальної SDN мережі.....	88
4.2. Обґрунтування вибору базового програмного забезпечення SDN-контролера .....	92
4.3. Програмна реалізація обчислення мінімального значення метрики якості обслуговування каналу зв'язку .....	95
4.4. Конфігурація SDN-контролера для проведення моделювання мережі .....	99
4.5. Порядок проведення моделювання.....	103
4.6. Побудова віртуальної мережі на основі технології SDN.....	107
4.6.1. Повнозв'язна топологія.....	109
4.6.2. Топологія де Бруйна.....	112
4.6.3. Топологія тор .....	115
4.7. Аналіз отриманих результатів моделювання.....	118
Висновки до розділу 4 .....	126
ВИСНОВКИ .....	128
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	130
ДОДАТОК А.....	141
ДОДАТОК Б .....	148

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – application programming interface, інтерфейс прикладного програмування

ARM – Advanced RISC Machine, вдосконалена RISC-машина

ARP – Address Resolution Protocol, Протокол вирішення адрес

CLI – command line interface, інтерфейс командного рядка

CMTS – cable modem termination system, система термінації кабельного модема

ICMP – Internet Control Message Protocol, Протокол керуючих повідомлень

Інтернету

MPLS – Multiprotocol Label Switching, Багатопротокольне перемикування міток

ODL – OpenDayLight

ONF – Open Networking Foundation

ONOS – Open Network Operating System, Відкрита мережева операційна система

OSGi – Open Services Gateway initiative

OSI – Open Systems Interconnection, Взаємозв'язок відкритих систем

SDN – software defined networks, програмно-визначені мережі

VLAN – Virtual Local Area Network, віртуальна локальна мережа

VPN – virtual private network, віртуальна приватна мережа

VXLAN – Virtual Extensible Local Area Network, Віртуальна розширювана  
локальна мережа

QoS – quality of service, якість обслуговування

REST – Representational State Transfer, Передача представницького стану

ЕБ – ексабайт

## ВСТУП

**Актуальність теми.** Практика використання комп'ютерних мереж останніми десятиліттями демонструє тенденцію до швидкого зростання кількості користувачів. Це призводить до підвищених ризиків для функціонування мережевої інфраструктури – перевантаження каналів зв'язку, падіння швидкості з'єднання та, як наслідок, відмови окремих вузлів чи усієї мережі в цілому. Окрім того, наявність людського фактору в налаштуванні мережі та необхідність окремої конфігурації для кожного вузла в системі значно погіршує гнучкість та масштабованість традиційних комп'ютерних мереж. Означені проблеми покликана вирішити технологія мереж, що програмно визначені (*SDN*).

Впровадження мереж *SDN* є актуальним рішенням, оскільки вони втілюють наступні інноваційні підходи до управління мережами. *SDN* дозволяє адміністраторам мережі здійснювати управління та конфігурацію мережевих пристроїв через так званий контролер. Коли технологія програмно-визначених мереж була вперше представлена, рівень обчислювальних потужностей не дозволяв здійснити її масове впровадження. Зараз, коли контролер може безпроблемно функціонувати на базі персонального комп'ютера або віртуальної машини, існує можливість для створення *SDN* мереж будь-якої розмірності та для передачі будь-якого типу трафіку. За рахунок централізованого управління вузлами і ресурсами контролеру вдається зменшити обсяг службової інформації, уникати зайвого використання пропускну здатності та оптимізувати шляхи передачі даних між вузлами. Порівняно з традиційними, мережі *SDN* значно ефективніше масштабуються, а за рахунок використання програмних інтерфейсів між контролером і пристроями мережі існує можливість задіяти велику множину налаштувань.

Для повного розкриття потенціалу використання мереж *SDN* актуальною є розробка комплексного рішення для поєднання технологій *SDN* та віртуальних мереж. Передумови використання віртуальних мереж полягають у збільшенні кількості мережевих пристроїв та зв'язків між ними. На відміну від фізичних,

віртуальні мережі при необхідності їхнього розширення не потребуватимуть переміщення та перепідключення пристроїв, що дозволить значно швидше масштабувати мережу: додати нові вузли та підключення між ними. Це добре погоджується з принципом мережі *SDN*, коли конфігурація пристроїв відбувається шляхом передачі команд з центрального контролера.

Актуальною проблемою також є можливість сучасних комп'ютерних мереж передавати різні типи трафіку. Так, існують види трафіку з різними вимогами до надійності – наприклад, трафік, пов'язаний з фінансовими операціями, виставляє більші вимоги до гарантій доставки інформації, ніж мультимедійний трафік. Для того, щоб відповідати вимогам, мережа мусить мати відповідну якість обслуговування (*quality of service, QoS*). Вона досягається шляхом підтримки певного рівня ряду метрик, як-от, кількість переходів між вузлами або затримка передачі даних. Щоб мати можливість ефективно передавати трафік різного типу, мережа має мати відповідність якомога більшому (не менше трьох-чотирьох) числу метрик якості обслуговування. Втім, більшість сучасних рішень побудови мереж зазвичай пропонують відповідність лише одній-двом метрикам *QoS*.

З огляду на означені проблеми, виникає необхідність в розробленні комплексного способу побудови комп'ютерної мережі, яке спрямоване на вдосконалення процесів передачі даних в комп'ютерних мережах та забезпечення необхідного рівня якості обслуговування завдяки інтеграції технологій віртуальних мереж і мереж *SDN*.

**Зв'язок роботи з науковими програмами, планами, темами.** Дисертаційна робота входить в план наукової роботи кафедри обчислювальної техніки КПІ ім. Ігоря Сікорського і виконана в рамках наступних пошукових досліджень (ініціативних тематик): «Високопродуктивні комп'ютерні системи та мережі: теорія, методи і засоби апаратної та програмної реалізації» (факультет інформатики та обчислювальної техніки – керівник: доц. А. М. Волокита), № договору: Д/р №0121U108261, дата реєстрації: 11.02.2021.

**Мета і завдання дослідження.** Метою дисертаційної роботи є підвищення швидкості та надійності передачі даних шляхом розробки комплексного способу

побудови комп'ютерної мережі, що дозволить гарантувати відповідність мережі заданим параметрам якості обслуговування під час виходу з ладу окремих її ділянок шляхом інтеграції технологій віртуальних мереж та програмно-визначених мереж (*SDN*).

Для досягнення мети в дисертаційній роботі поставлені наступні завдання:

- Дослідити аспекти побудови, а також принципи функціонування віртуальних мереж з використанням технології *SDN* з метою розроблення комплексного способу побудови комп'ютерної мережі, що дозволить вдосконалити процеси передачі даних у випадку відмов окремих ділянок комп'ютерної мережі та забезпечення необхідних параметрів якості обслуговування;
- Розробити метод побудови каналів зв'язку між вузлами мережі, який враховує необхідну кількість метрик якості обслуговування каналів зв'язку з метою збереження швидкості, зменшення затримки та втрат даних при передачі різних типів трафіку у разі виходу з ладу окремих ділянок мережі;
- Вдосконалити метод динамічної реконфігурації віртуальної комп'ютерної мережі на основі технології *SDN*, який дозволяє підтримувати сеанс передачі трафіку у разі відмови основного каналу зв'язку, використовуючи для цього резервні канали зв'язку між вузлами;
- Обґрунтувати технічне рішення для *SDN* контроллера з метою реалізації запропонованих методів побудови каналів зв'язку віртуальної *SDN* мережі з урахуванням метрик якості обслуговування та динамічної реконфігурації мережі з використанням резервних каналів зв'язку;
- Розробити програмне забезпечення для проведення експериментів з метою доведення ефективності запропонованого комплексного способу побудови віртуальної комп'ютерної мережі на основі технології *SDN*.

**Об'єкт дослідження.** Об'єктом дослідження дисертаційної роботи є процес побудови віртуальної комп'ютерної мережі на основі технології *SDN* з використанням іноваційних методів передачі даних.

**Предмет дослідження.** Предметом дослідження дисертаційної роботи є



методи та засоби вдосконалення процесів передачі даних у віртуальних комп'ютерних мережах на основі технології *SDN*.

**Методи дослідження.** Методичною основою дослідження є системне опрацювання та аналіз теоретичного матеріалу, присвяченого віртуальним комп'ютерним мережам, програмно-визначеним мережам, процесам передачі даних у них, алгоритмам маршрутизації, принципам функціонування *SDN* контролерів.

У процесі виконаного дослідження були використані методи порівняльного та статистичного аналізу, метрики обчислювальної складності, швидкості та кількості передачі даних у комп'ютерних мережах, а також програмні засоби для моделювання віртуальних мереж та функціонування *SDN* контролерів.

**Наукова новизна отриманих результатів.** Наукова новизна одержаних результатів роботи полягає у наступному:

- Вперше розроблено та обґрунтовано інтегральний критерій побудови каналів зв'язку у віртуальній комп'ютерній мережі на основі технології *SDN*, який, на відміну від існуючих способів побудови, одночасно враховує необхідну кількість та пріоритети метрик якості обслуговування каналів зв'язку мережі, що дозволяє досягти збереження стабільної швидкості передачі трафіку будь-якого типу;
- Вперше розроблено та обґрунтовано критерій надійності вузлів мережі для побудови каналів зв'язку, який, на відміну від існуючих способів побудови, враховує здатність проміжних пристроїв мережі транспортувати трафік між своїми портами, що дозволяє збільшити надійність каналів зв'язку та зменшити втрати даних під час передачі;
- Вперше розроблено та обґрунтовано метод побудови відмовостійких каналів зв'язку віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на розділенні віртуальної мережі на підмережі та використовує розроблені інтегральний критерій побудови каналів зв'язку з урахуванням якості обслуговування та критерій надійності вузлів, що дозволяє забезпечити стабільну швидкість, зменшити

затримку та втрати даних під час передачі будь-якого типу трафіку у разі виходу з ладу окремих ділянок мережі;

- Удосконалено метод динамічної реконфігурації віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на використанні відмовостійких резервних каналів зв'язку, збережених у пам'яті контролера *SDN*, що дозволяє підтримувати безперервний сеанс передачі трафіку у разі відмови основного каналу зв'язку.

**Практичне значення отриманих результатів.** Запропоновані методи та одержані результати дозволяють реалізувати на практиці комплексне рішення для побудови віртуальної комп'ютерної мережі на основі технології *SDN*. Запропоновані методи реалізовані у вигляді програмного рішення, що представляє собою централізований *SDN* контролер на базі програмного забезпечення *Open Network Operating System (ONOS)*, та модифікований застосунок *Intent Forwarding*, який виконує маршрутизацію між *OpenFlow*-комутаторами у мережі. Розроблене комплексне рішення дозволяє досягти переваг у швидкості реконфігурації мережі, підвищенні надійності передачі даних у порівнянні з існуючими практичними рішеннями побудови віртуальних мереж на основі технології *SDN*.

**Особистий внесок здобувача.** Дисертація є результатом самостійних наукових досліджень, у яких викладено авторські наукові досягнення у побудові віртуальних мереж на основі технології *SDN*. Наукові положення та основні результати, які містяться в дисертації, отримані здобувачем самостійно у процесі науково-дослідницької роботи. В роботах, опублікованих у співавторстві, дисертанту належать: [1] – розробка методу побудови віртуальної комп'ютерної мережі на основі технології *SDN* з використанням алгоритму реконфігурації з резервними маршрутами та аналіз завантаження каналів зв'язку при відмовах проміжних *OpenFlow* комутаторів; [2] – розробка методу побудови віртуальної мережі *SDN* під управлінням централізованого контролера на базі програмного забезпечення *ONOS* з підтримкою протоколу *OpenFlow*, який виконує конфігурацію та підтримку мережі і реалізує механізм реконфігурації віртуальних комутаторів; [3] – проведення та опис тестування представлених топологічних

організацій мережі у віртуальному середовищі *SDN* на відмовостійкість шляхом експериментального виміру об'єму переданого трафіку після відключення частини вузлів та виявлення найбільш стабільних до змін типів топологічної організації комутаторів; [4] – опис методу підвищення стабільності передачі даних у програмно-визначеній мережі (*SDN*) шляхом її динамічної реконфігурації з урахуванням параметрів якості обслуговування (*QoS*) каналів зв'язку та надійності вузлів; [5] – порівняльний аналіз методу конструювання трафіку зі спеціалізованими параметрами якості обслуговування для топологічних організацій віртуальної мережі *SDN* та графічне представлення результатів експерименту.

**Апробація результатів дисертації.** Основні результати роботи опубліковано та обговорено на XV Міжнародній науково-практичній конференції «Комп'ютерні системи та мережні технології» (м. Київ, Україна, квітень 2024) та міжнародній конференції The International Conference on Security, Fault Tolerance, Intelligence 2024 (м. Київ, Україна, червень 2024).

**Публікації.** За результатами дисертаційних досліджень у фахових українських та міжнародних виданнях опубліковано 5 наукових статей, з яких 2 входять до наукометричної бази даних з міжнародним індексом цитування *Scopus*.

**Структура і обсяг роботи.** Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 90 найменувань. Загальний обсяг дисертації становить 149 сторінок, з яких 128 сторінки основного тексту, 2 додатки на 9 сторінках. Дисертація містить 45 рисунків, 36 формул, 12 таблиць.

## РОЗДІЛ 1

### АНАЛІЗ ВІДОМИХ РІШЕНЬ ПОБУДОВИ ПРОГРАМНО- ВИЗНАЧЕНИХ МЕРЕЖ

#### 1.1. Проблема зростання об'єму трафіку у комп'ютерних мережах

Не дивлячись на те, що комп'ютерні мережі стають важливою частиною технологічного прогресу, багато сучасних мережевих інфраструктур все ще ґрунтуються на раніше впроваджених архітектурних рішеннях ХХ століття, які стрімко втрачають актуальність. Сьогодні спостерігається нездатність цих рішень побудови мереж адекватно реагувати на динамічні умови: наприклад, з ростом кількості і різноманітності мобільних пристроїв, бездротовий зв'язок швидко прогресує, виникають нові технології і стандарти для нього. Це також призводить до швидкого зростання кількості користувачів Інтернету, так як висока доступність мобільних пристроїв дозволяє збільшити зручність підключення до світової павутини. Як результат, сучасні бездротові мобільні мережі стикаються з більшими викликами, ніж традиційні провідні інфраструктури.

Основною проблемою для операторів зв'язку є стрімке зростання обсягів мобільного трафіку. Окрім збільшення кількості кінцевих користувачів Інтернету, мобільні технології і додатки стають все більш високопродуктивними, а значить зростає пропускна здатність, яка вимагається від мереж. Види і об'єм трафіку, що передаються мережами, розширюються. Регулярно оновлюються стандарти мультимедійних форматів та якості контенту, що збільшує навантаження на ширококомовні комп'ютерні мережі.

У 2010-х та 2020-х роках проблема загострилась максимально з популяризацією відеохостингових та стрімінгових платформ, а пандемія *COVID-19* викликала різкий ріст Інтернет-трафіку через перехід більшості сервісів на віддалений режим роботи і зв'язку. Статистичні дані демонструють, що об'єм інтернет-трафіку по світу стабільно зростає на 10-15% щороку, а відсоток мобільного трафіку 2023 року вперше сягнув 60% (табл. 1.1, 1.2) [1].

Таблиця 1.1

## Об'єм світового Інтернет-трафіку по рокам

Квартал	Об'єм трафіку, ЕБ	Квартал	Об'єм трафіку, ЕБ
Q1 2019	29,02	Q2 2021	72,2
Q2 2019	32,66	Q3 2021	77,81
Q3 2019	36,49	Q4 2021	84,16
Q4 2019	39,56	Q1 2022	92,75
Q1 2020	45,16	Q2 2022	100,42
Q2 2020	50,03	Q3 2022	107,58
Q3 2020	54,79	Q4 2022	118,21
Q4 2020	58,44	Q1 2023	126
Q1 2021	66,14	Q2 2023	133,86

Таблиця 1.2

## Відсоток мобільного трафіку у світі

Квартал	Відсоток мобільного трафіку
Q1 2015	31,16%
Q1 2016	39,47%
Q1 2017	50,03%
Q1 2018	51,77%
Q1 2019	48,71%
Q1 2020	51,92%
Q1 2021	54,8%
Q1 2022	59,66%
Q1 2023	60,89%

Для розв'язання проблеми перевантаження мереж пропонувались різні варіанти. Один з тих, що отримав велику підтримку – збільшення густоти мобільного покриття за допомогою збільшення числа базових станцій та зменшення розміру території, яку вони покривають. Завдяки цьому клієнти будуть

розташовані ближче до станцій, а їхня кількість на одну станцію скоротиться. Але для досягнення ефективності такого рішення, за профільними оцінками, потрібно майже вдвадцятьох збільшити кількість базових станцій [2].

Проте, існують причини, через які проблеми поточної мережевої архітектури не будуть вирішені простим збільшенням кількості сот у мобільних мережах:

- Неможливо однорідно розширити мобільне покриття, оскільки базові станції будуть розташовуватися випадковим чином через різні рельєфні умови та різне споживання послуг користувачами (наприклад, при порівнянні міської та сільської зон);
- Керування такою великою мережею буде дуже складним і незручним через навантаження, що хаотично змінюватиметься, взаємодії між сотами та багато інших факторів;
- Для забезпечення більш щільного покриття необхідна велика кількість обладнання, що призведе до значних витрат на впровадження такої інфраструктури та її подальше обслуговування.

Іншим рішенням проблеми зростаючого трафіку може бути оновлення обладнання самих базових станцій задля збільшення їхньої пропускної здатності. Оскільки у світі щороку створюються нові специфікації та інновації, існує можливість за рахунок вдосконалення апаратного забезпечення вирішити проблему перевантаження мереж. Однак, немає впевненості, що апаратна складова пристроїв та їх програмне обладнання працюватимуть відповідно до документації – як немає такої впевненості і для діючого обладнання – тому можуть виникнути непередбачені обмеження функціоналу, відмінні від того, що описано в документах. У мережах ця проблема стає навіть більш важливою, оскільки функції мережевих пристроїв розподілені по вузлах з'єднань. Основною причиною цього є домінування ліцензованого пропрієтарного програмного та апаратного забезпечення. Зазвичай, виробники вкладають директиви для адміністрування мережею в апаратну частину своїх пристроїв, не задіюючи програмний рівень, тому без участі самого виробника (який мусить мати до цього відповідну

фінансову або технічну мотивацію) перепроєктувати вузли існуючих мереж є вкрай складною задачею.

При покращенні обладнання та інфраструктури варто враховувати, що за останні десять років організація обчислювальних процесів поступово переміщується до хмар (*cloud*), а клієнт-серверна архітектура з фізичним сервером, що виконував необхідні операції, яка була основою для всіх комп'ютерних мереж раніше, нині втрачає позиції на фоні розвитку віртуальних мереж [3]. Для того, щоб реалізувати впровадження цих технологій, рівень розвитку мережевої інфраструктури, що планується для побудови, має бути достатньо сучасним.

Іншим фактором, що викликає труднощі у функціонуванні мереж на нинішньому етапі розвитку мережевих технологій, є ускладнення процесу управління. Оскільки кількість мереж зростає, то вимоги до безпеки та надійності мають бути забезпечені через оновлення систем безпеки, мережевих протоколів та устаткування у більшому масштабі, ніж раніше. Мережеві пристрої, на основі яких створюються мережі, стають все більш складними і мають підтримувати розподілені функції – наприклад, нові протоколи, кількість яких (як і їхніх стандартів) постійно зростає. При цьому виробники включають у обладнання свої власні пропріетарні інтерфейси, які, зазвичай, мають обмежену підтримку протоколів, особливо тих, що закриті від інших виробників. У постачальників послуг можливості щодо модернізації архітектури своїх інфраструктур, впровадження нових сервісів, додавання нових клієнтів, що постійно збільшують свої вимоги, є дуже обмеженими. Кожна метрика мережі, як-от, швидкість, надійність, кількість користувачів, вимагає постійного моніторингу та удосконалення, щоб задовольняти постійно зростаючі вимоги користувачів та сучасних технологій. В результаті, керування сучасною мережевою інфраструктурою передбачає не тільки виконання широкого спектру технічних завдань, але і створення ефективної стратегії, яка допомогла б керувати ростом мережі і ефективно реагувати на зміни в потребах користувачів.

Таким чином, вже на початок 2010-х років проблема збільшення пропускної спроможності мереж стала критичною через перевантаження більшості каналів у

зв'язку з вичерпанням їхніх апаратних спроможностей при обмежених можливостях до оновлення конфігурацій. Відповідно, швидкість розширення мереж та їхня пропускна здатність не може задовольняти зростаючі потреби користувачів, а тому будь-які можливості, що дає традиційний підхід до створення мереж, не можуть розв'язати наявних проблем [4].

Переконливим свідченням деякої «кризи» у сфері управління мережами глобального масштабу є велика кількість кібератак на інфраструктуру і масових вірусів, що негативно впливають на економічну активність в багатьох країнах. Оскільки ураження мереж продовжується регулярно, це свідчить про те, що проблема безпеки мереж, яка актуальна для їхньої підтримки протягом останніх десятиліть, все ще залишається нерозв'язаною [5].

Україна має унікальний набір проблем, пов'язаних з розвитком інфраструктури мережевих провайдерів. Оскільки у 70-80-х роках використовувались стандарти побудови мережевої архітектури, які не мали відношення до західних аналогів, то на початок XXI століття в Україні не сформувалось свого комплексу стандартів і виробників мережевого обладнання. У нашій державі не існує вітчизняного виробництва технологій, що стосуються Інтернет-трафіку, і вся інфраструктура створюється на основі устаткування від іноземних виробників. В цьому контексті Україна залежить від міжнародних стандартів і трендів в мережевій індустрії.

Отже, виокремимо головні труднощі сучасних комп'ютерних мереж, які вимагають переходу на програмне конфігурування:

- технічні обмеження та наукова складність: нинішні глобальні комп'ютерні мережі стали досить складними для ефективного керування, забезпечення їх безпеки та прогнозування їхньої поведінки;
- економічний аспект: мережева архітектура дорожчає та стає більш заплутаною, що вимагає залучення більш кваліфікованих спеціалістів для управління мережами та придбання дорогого обладнання;



- питання майбутнього: наявні рішення мережевої архітектури обмежують можливості масштабування та додавання нових сервісів для обслуговування.

Тож підсумуємо, що питання масштабованості мереж для збільшення їхнього покриття та кількості користувачів набуло критичної актуальності. Величезне розмаїття технічних засобів зв'язку, які з'явилися з початку ХХІ століття, вимагає у системних операторів знання їхньої архітектури та команд для керування, які розрізняються від одного виробника до іншого. Ці виклики спонукають до виникнення необхідності у впровадженні повної або часткової автоматизації даного процесу.

## **1.2. Концептуальний аналіз технології віртуальних мереж**

*Wi-Fi* та мережі мобільних даних працюють на основі бездротових радіосигналів. В семишаровій моделі *OSI* встановлення зв'язку та маршрутизація між пристроями знаходяться на трьох найнижчих шарах: 3-й рівень (мережевий), 2-й рівень (канальний) і 1-й рівень (фізичний).

У віртуальних мережах зв'язок та маршрутизація повністю відбуваються в програмному середовищі, яке може функціонувати всередині одного фізичного сервера. Або ж, вони можуть бути рівнем абстракції, що діє поверх фізичної мережі, де налаштування та структура можуть значно відхилитися від налаштувань та структури віртуальної мережі. Це збільшує гнучкість побудови інфраструктури для мобільних мереж [6].

Створення віртуальної мережі – це складний процес, але переваги його використання доволі значні: мережі можна змінити, просто редагуючи файл, без потреби в важкій фізичній роботі по перепідключенню пристроїв, замість яких тут виступають віртуальні машини.

Віртуальна машина – програмна сутність, яка дає змогу використовувати декілька програм на одному фізичному обладнанні. Усі системні запити та іншу діалогову діяльність, яку вони зазвичай проводять із базовим апаратним

забезпеченням, керує програмний рівень, який називається гіпервізор. Він керує запитами від різних віртуальних систем, які працюють на одній фізичній машині, для оптимального використання базових апаратних ресурсів [7].

Віртуальна машина може адаптувати команди між апаратними платформами, що дозволяє їй, наприклад, працювати на обладнанні x86, незалежно від того, чи була вона створена для процесорів ARM. Гіпервізор при цьому здатен надавати відповіді, які віртуальна машина очікує отримати від апаратного обладнання.

Віртуальна мережа побудована на аналогічних засадах. Вона використовує програмне забезпечення для імітації специфічної мережевої структури за вимогою мережевого адміністратора. Подібно до віртуальних машин, віртуальну мережу описує програмне забезпечення, а не реальне обладнання. Це означає, що системи або програми, які використовують цю мережу, не розрізняють, чи вони пов'язані з реальним комутатором, чи віртуальним [8].

Мережеві пристрої відправляють пакети зі спеціальною інформацією для маршрутизації в заголовках і очікують отримати їх назад. Через стандартизацію такого зв'язку можна з легкістю створити програмне забезпечення, яке імітує роботу фізичної мережевої карти, комутатора або роутера. Такі віртуальні комутатори, як *Open vSwitch*, мають можливість функціонувати на гіпервізорі або як керівний шар для реального мережевого обладнання.

З отриманими пакетами гіпервізор повинен визначити шлях до місця призначення згідно з тим, як це б здійснювалося у фізичній мережі. Однак відмінність полягає в тому, що гіпервізор повинен перетворити програмну інформацію з віртуальної мережі на об'єктивні дані про реальне фізичне середовище [9].

Гнучкість віртуальної мережі для побудови мобільних інфраструктур полягає в тому, що хост може вважати, що він відправляє пакет до іншого хоста в межах тієї ж локальної мережі, проте насправді ці два пристрої можуть бути розташовані у різних містах чи країнах.

Гіпервізор розв'язує цю проблему, розміщуючи пакет всередину іншого пакета, який має інформацію про маршрутизацію в заголовку, а після цього

передає цей складений пакет до реального мережевого обладнання. Коли цей пакет доходить до свого призначення, зовнішній пакет видаляється, а пристрій, який отримав його, буде сприймати пакет так, наче він прийшов через віртуальну мережу, а не через реальну фізичну мережу, по якій він насправді проходив [10].

Віртуальна приватна мережа (*VPN*) є найбільш базовим типом віртуальних мереж. Найбільш типова ситуація її використання – коли деякий комп'ютер підключається через Інтернет до корпоративної локальної мережі. Для цього комп'ютера і інших комп'ютерів у мережі, з якими він взаємодіє, цей комп'ютер буде виглядати як частина локальної мережі після підключення до *VPN*, незалежно від його географічного розташування.

Віртуальна локальна мережа (*VLAN*) формує повністю локальну мережу, що існує у віртуальному середовищі. *VLAN* може бути створена шляхом поділу однієї фізичної локальної мережі на декілька *VLAN* або злиття фізично відокремлених локальних мереж в одну спільну *VLAN*.

Віртуальна розширена локальна мережа (*VXLAN*) представляє собою вдосконалену версію *VLAN*, що дозволяє розділити великі локальні мережі на низку окремих *VLAN*. Вона дозволяє мігрувати конфігурацію віртуальних машин без переривання послуг, що є важливим для хмарних середовищ. *VXLAN* включає більше мережевих можливостей, що робить її дуже ефективним рішенням для управління великими та складними віртуальними мережами [11].

Віртуальна мережа стає незамінною для використання, коли існують декілька віртуальних машин, що працюють на одній фізичній платформі, що є звичайним випадком у сфері хмарних обчислень. Через створення мереж *VLAN* і *VXLAN* в межах існуючих реальних мереж, мережеві адміністратори можуть швидко перебудовувати мережі для відповіді на потреби – це набагато ефективніше, ніж переорганізувати реальні мережі вручну і є дуже важливою особливістю для побудови мобільних мереж *SDN*, про які піде мова у наступних пунктах.

Отже, основні переваги віртуальних мереж, що роблять їх найбільш підходящою технологією для побудови мобільних програмно-визначених мереж є: зниження витрат на обслуговування фізичного мережевого обладнання,

спрощення керування мережею через централізацію і автоматизацію управління, більш гнучкі та деталізовані опції налаштування мережі.

### 1.3. Аналіз архітектурних особливостей мереж SDN

Вперше технологія *SDN* була представлена у 2006 році, будучи створена дослідниками з Університету Стенфорда та Університету Каліфорнії в Берклі. Хоча наукове середовище відразу ж оцінило нову концепцію як проривну, великі технологічні корпорації не проявляли до неї великого зацікавлення протягом найближчих років. Лише у 2011 році був заснований консорціум *Open Networking Foundation (ONF)*, на якому було узгоджено спільну працю над вдосконаленням та розробкою автоматизованих мереж нового покоління. Цей консорціум об'єднав великі компанії, які спеціалізуються на виробництві мережевого обладнання, як-от *T-Mobile, Microsoft, Google, Intel, AT&T* [12].

Головною ідеєю, що закладається у мережу, що програмно визначена, є відокремлення рівнів передачі й адміністрування трафіку у мережі (рис. 1.1) [13].

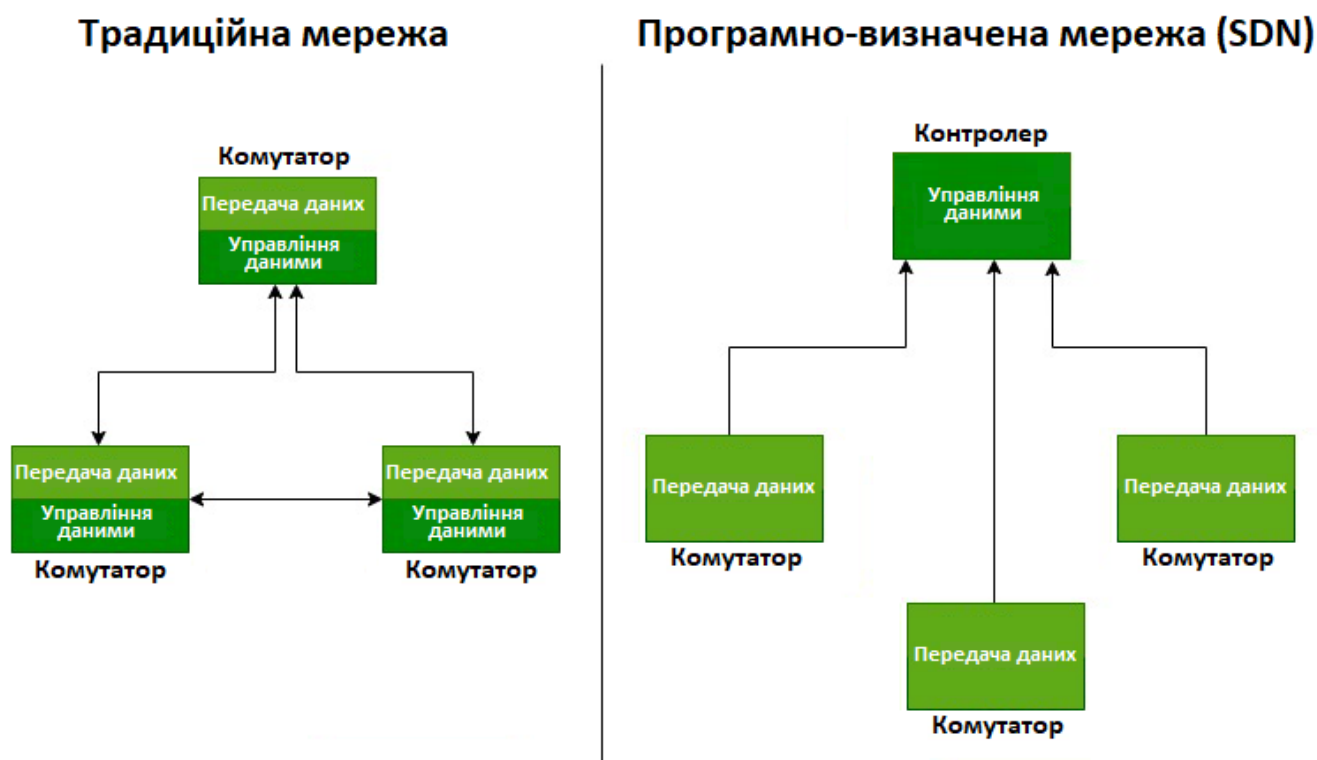


Рис. 1.1. Візуалізація рівнів передачі даних (*data plane*) та управління даними (*control plane*) у традиційній та *SDN* мережі

Впровадження *SDN* мереж має на меті виправити принаймні частину накопичених проблем управління, за допомогою автоматизації, вдосконалення безпекових стандартів та збільшення гнучкості у функціонуванні та управлінні комп'ютерними мережами.

Розділення рівнів передачі й адміністрування дозволяє забезпечити динамічну конфігурацію мережі для покращення її продуктивності та моніторингу, у спосіб, більш схожий на хмарні обчислення, ніж на традиційне керування мережею [14]. Така ідея дозволяє вирішити проблему статичної архітектури традиційних мереж і може використовуватися для централізації адміністрування в одному мережевому компоненті (центральному контролері), який працює у площині управління даними (площина керування, *control plane*), в той час як пристрої мережі залишаються лише в площині пересилання даних (площина даних, *data plane*). Тому, на відміну від традиційної мережі, де кожний пристрій у її складі відповідає як за передачу даних, так і за маршрутизацію по мережі через «спілкування» з іншими ланками, в програмно-визначеній мережі усі проміжні пристрої обмінюються директивами лише з контролером і пересилають дані через себе на основі його команд [15].

Відповідним чином може бути змінена апаратна і програмна конструкція пристроїв у ланках мережі, які при традиційному підході до побудови інфраструктури виконують одразу три логічні функції: передачу трафіку, керування трафіком та управління мережевими пристроями (в деяких публікаціях це виділяється в окрему площину адміністрування – *management plane* [16]). Зазвичай, кожний з цих процесів контролюється окремо на кожному елементі мережі, що призводить до значних затрат часу на конфігурацію мережі і неефективного використання ресурсів. В *SDN* мережі проміжні комутатори будуть складатися тільки з двох частин – фізичних мікрочіпів, відповідальних за можливість передачі трафіку, та базової таблиці маршрутизації, що надсилається пристрою центральним контролером та вказує пакетам напрямок переміщення між портами пристрою (рис. 1.2) [17].

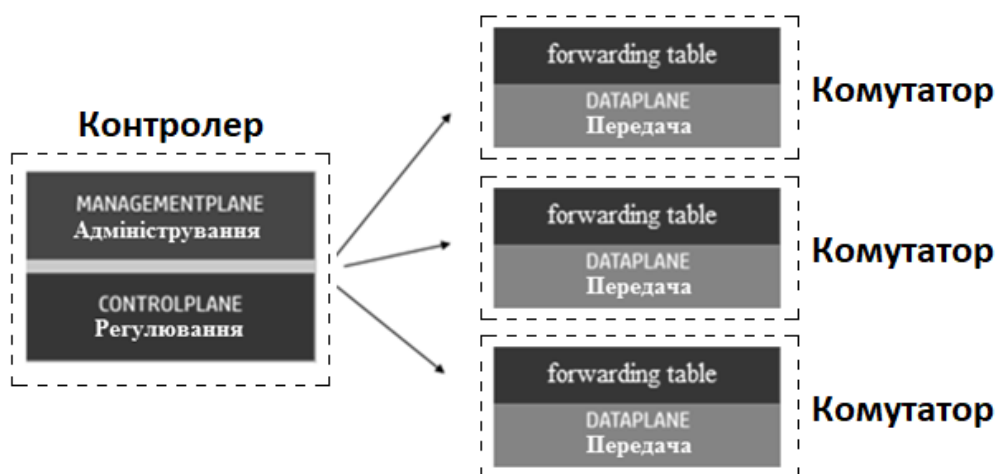


Рис. 1.2. Об'єднання площин регулювання та адміністрування і розділення з площиною передачі даних у мережах *SDN*

Розділення площин керування та управління даними в *SDN* мережі та передача адміністрування в окрему централізовану систему, яка автоматизованим чином буде одночасно реконфігурувати всю мережу, дозволяє зменшити витрати часу на процес ініціалізації налаштувань та збільшити продуктивність передачі трафіку [18].

Привабливість цієї технології для корпоративних мереж полягає в наступних фактах:

- можливість покращити ефективність роботи мережевого обладнання і знизити витрати на його обслуговування, завдяки обмеженню можливості людської помилки у адмініструванні та швидкому виявленню і виправленню помилок;
- перетворення управління мережею на інженерне рішення;
- можливість створення спеціалізованих сервісів для адміністраторів і кінцевих користувачів, які можуть бути миттєво завантажені або видалені за рішенням центрального контролера, що спрощує доступ та контроль за мережею [19].

Технологія *SDN* представляє інноваційний підхід до мереж, який обходить складність та статичність децентралізованих мережевих структур, що переважно використовуються сьогодні, за допомогою програмного підходу, побудованого на

принципі абстракції площини керування та площини передачі даних, що включає як фізичні, так і віртуальні пристрої. У комплексі це абстрагування створює нові стандарти мережевого управління, що дозволяють виконувати динамічне адміністрування мережі з центрального контролера за допомогою програмного забезпечення.

Зокрема, за допомогою протоколу абстракції площини даних, такого як *OpenFlow*, можна створювати мережу з будь-яким типом мережевого обладнання і виробника, адже увесь базовий набір пристроїв може зв'язуватись між собою, використовуючи цей загальний протокол абстракції. Це усуває традиційні обмеження, що зазвичай накладаються виробниками на свої пристрої, надаючи більше гнучкості в налаштуваннях для програмістів та адміністраторів [20].

Схематичне представлення інформації про структуру архітектури *SDN* наведено на рисунку 1.3 [21].

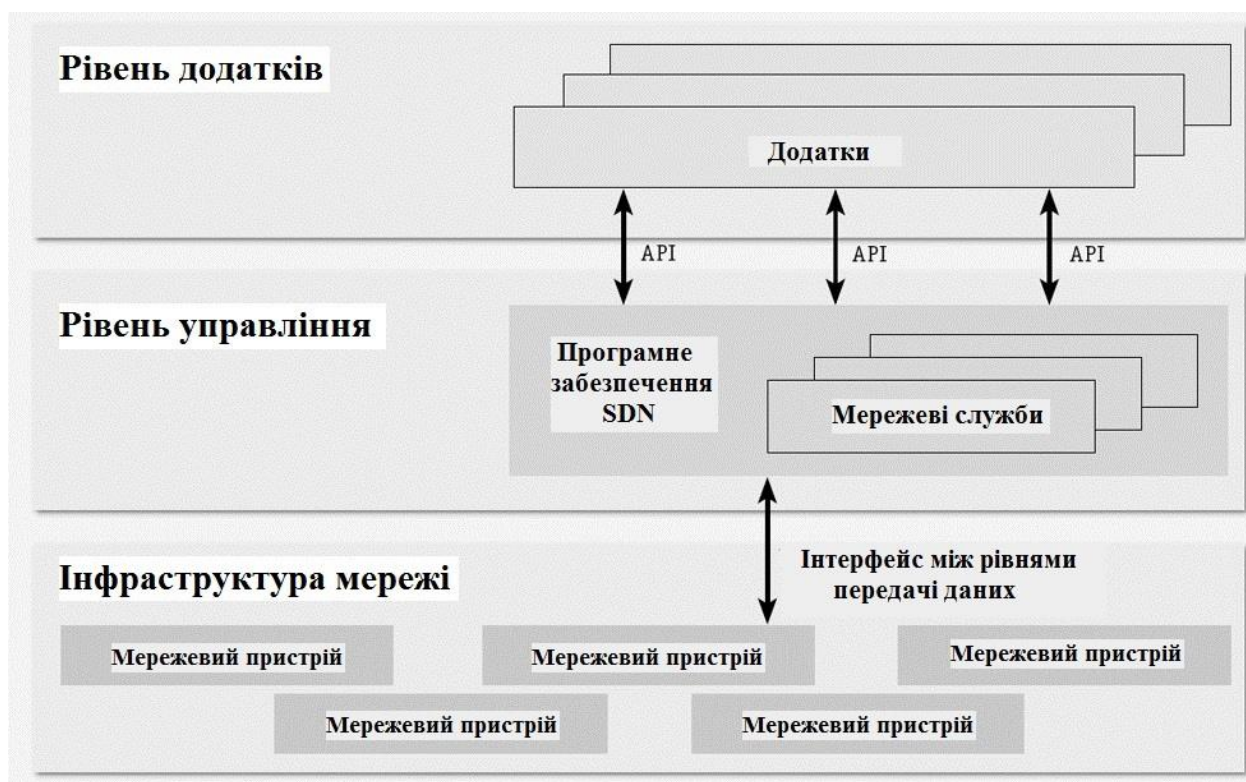


Рис. 1.3. Схема взаємодії рівнів у архітектурі мережі *SDN*

Комунікація між *SDN* контролером та проміжними пристроями у мережі відбувається за допомогою програмного інтерфейсу (*API*), який дозволяє їм обмінюватись великою кількістю спеціалізованих даних, що було неможливим у

традиційних мережах до виникнення *SDN*. При цьому, контролер не обмежений лише одним *API*, а може бути розширений й іншими інтерфейсами для досягнення певних можливостей, наприклад, пришвидшення взаємодії між пристроями або передачі даних у форматі, більш зрозумілому для людини. В останньому випадку, це спростить розробку і підтримку програмного забезпечення для контролера.

Порівняно з традиційними розподіленими підходами до створення мереж, які мають повну залежність від обладнання та стандартів, які у нього закладає виробник, *SDN* представляє собою більш сучасний підхід до управління нею та відкриває можливості по збільшенню швидкості та якості передачі трафіку, які були недосяжними раніше (табл. 1.3) [22].

Таблиця 1.3

Порівняльний аналіз традиційних та *SDN* мереж

Ознака	Традиційні мережі	Мережі <i>SDN</i>
Гнучкість	Мережа володіє гнучкістю та оперативністю відповідно до вимог.	
Рівні управління	Рівень передачі даних та рівень адміністрування поєднані в одному апаратному пристрої.	Рівень передачі даних та рівень адміністрування роз'єднані та спілкуються через виділений протокол (найчастіше – <i>OpenFlow</i> ). Рівень адміністрування централізований у контролері.
Відмовостійкість	Через ручну конфігурацію комутаторів мережа схильна до помилок.	Мережа централізовано керується та автоматично конфігурується відповідно до вимог.
Централізація	Відсутнє поняття централізованого огляду мережі.	Контролер має централізоване представлення мережі.



Продовження таблиці 1.3

Складність управління	Управління мережею дуже складне.	Управління мережею просте завдяки розгортанню контролерів.
Масштабування	Масштабування мережі неможливе ефективним способом.	Завдяки роз'єднанню рівнів адміністрування та передачі, масштабування мережі легке.
Вартість управління	Вартість управління мережею висока.	Вартість управління мережею нижча.
Тип налаштування	Складається з апаратних мережевих пристроїв.	Налаштовується на основі програмного забезпечення з відкритим кодом.

#### 1.4. Аналіз відомих наукових підходів до побудови мереж SDN

В останні роки був опублікований цілий ряд досліджень про побудову програмно-визначених мереж, але існують певні нерозкриті можливості та невирішені проблеми у контексті проектування цих мереж.

У публікації [23] розглядається проблема оновлень програмного забезпечення *SDN*-контролера та ризиків перебоїв у роботі мережі. Для реконфігурації зазвичай потрібен рестарт контролера, або використання резервних систем та спеціалізованого ПО, що не є допустимим для більшості корпоративних мереж. В статті автори розглядають способи вирішення цих викликів для *SDN* за допомогою динамічної реконфігурації і демонструють як можливо контролювати стан мережі під час цього процесу. Однак розроблений ними метод використовує лише одну центральну інстанцію для керування динамічною реконфігурацією, у той час як для масштабних інфраструктур для зменшення часу простою та збільшення відмовостійкості потрібна розподілена система контролерів.

Біанко та інші [24] висвітлюють проблему масштабування мереж *SDN*. Такі чинники, як значна географічна розподіленість і потреба в передачі контрольного трафіку, викликають додаткові складнощі при використанні *SDN* в мережах Інтернет-провайдерів. У своєму дослідженні автори розробили кількісну модель для розрахунку величини обміну повідомленнями *OpenFlow* і відповідної пропускної здатності, необхідної для передачі трафіку при використанні алгоритмів пересилання з контролера *ONOS*. Однак, хоча стверджується, що розроблені кількісні моделі можуть слугувати інструментом для надійного планування мережі, авторами не проведений аналіз безпеки такого рішення, оскільки про мережу збирається доволі детальна і, можливо, конфіденційна статистика.

У статті [25] пропонується нове мережеве рішення побудови під назвою *Open Network Hypervisor (ONVisor)* на основі *ONOS*. Основні можливості *ONVisor* включають окрему реалізацію площини управління та обміну даними для кожної віртуальної мережі, підтримку розподілених операцій, підтримку різноманітних типів даних для *SDN*. Ряд експериментів були проведені у різних тестових середовищах на основі різних сценаріїв та порівняні з невіртуалізованою *SDN* мережею. Втім, результати показали, що *ONVisor* надає перевагу у продуктивності функціонування площини керування, але не виграє у швидкості передачі даних, тому не вирішує поставлених задач комплексно.

У дослідженні [26] порівнюється затримка реконфігурації для централізованої (*SDN*) та розподіленої (*OSPF*) мережі. Результати моделювання вказують на те, що мережа *SDN* в цілому переважає традиційні у швидкості передачі даних, коли розглядаються такі фактори як розмір мережі, затримки з'єднань та локалізація збою. Але суттєвим недоліком дослідження є відсутність урахування при порівнянні різних рішень *SDN* контролерів, яких на момент публікації цієї роботи вже існували десятки.

Санвіто та інші [27, 28] опублікували серію наукових досліджень, де запропонували сервіс під назвою *Intent Monitor and Reroute (IMR)*, який розширює функціонал *Intent Framework* у контролері *ONOS*, надаючи йому можливість

одночасно компілювати декілька намірів (*intentions*) і перенастроювати потоки даних залежно від стану мережі та статистики потоку за допомогою команд, незалежних від платформи. Хоча рішення було визнане розробниками *ONOS* та включене до стандартного пакету додатків контролера, воно має ряд недоліків, як-от, відсутність підтримки багатошляхової маршрутизації та урахування затримки пакетів при передачі.

Політику намірів у *ONOS* вивчає й ряд інших досліджень. Так, у [29, 30] завдяки високій гнучкості та меншій затримці, ніж при пересиланні пакетів *ONOS* за замовчуванням, використання додатку *Intent Forwarding* дозволило збільшити швидкість пересилання пакетів на основі врахування пропускну здатності та використання мережевого трафіку, а не лише кількості переходів. У публікації [31] було продемонстроване прикладне використання програмно-визначеної мережі з використанням намірів *ONOS*, де контролер виконував задачі маршрутизації трафіку та його перенаправлення, щоб проаналізувати різницю у наборі даних, створених контролером *SDN*. Але наразі немає наукових досліджень, що дозволяли би використовувати політику намірів для пересилання даних до кількох адресатів одночасно, тобто позбулись би обмежень, які накладає на маршрутизацію алгоритм пошуку шляхів Дейкстри.

Побудову мережі на базі *SDN* з контролером, який взаємодіє з усіма комутаторами через спеціалізоване дерево, описує стаття [32]. В ній вирішується завдання знаходження мінімального дерева за вагою, де вага визначається як сума ваг усіх комутаторів, а саме дерево будується запропонованим евристичним алгоритмом, який мінімізує не лише вагу дерева, а й середню відстань між контролером та комутатором. Однак, такий підхід є лише розвитком ідеї обчислення метрики відстаней за кількістю переходів та при прикладній реалізації зіштовхнеться з неврахованими затримками та перевантаженням каналів.

Ряд публікацій вивчають проблему балансування завантаженості каналів у *SDN* мережах. Так, у статті [33] для вирішення проблеми перевантаженості пропонується комбінація стратегії вагового планування і динамічного перемикавання маршруту, що формує новий алгоритм балансування навантаження.

Він використовує хешування інформації, що сприяє зниженню затримок, а також забезпечує певний ступінь безпеки. Тим не менш, алгоритм був протестований лише на одній доволі простій топології, і для мереж великої розмірності зі складним топологічним графом його ефективність не була доведена.

Оригінальний підхід до вирішення проблеми врахування завантаженості каналів у *SDN* мережі був запропонований у статті [34]. При появі проблемної ділянки і необхідності її обходу використовується теорія прийняття рішень в умовах невизначеності, оскільки при виборі найкращого варіанта обходу враховується характер передачі трафіку. Застосування такого підходу дозволяє знизити втрати нееластичного трафіку, який становить значну частину загального обсягу даних. Далі ця ідея була розвинута у праці [35], де запропонований спосіб зменшення ймовірності динамічної реконфігурації шляхом прогнозування завантаження каналів у транспортній мережі. Контролер *SDN* генерує шляхи, що не перетинаються, за допомогою алгоритму зворотної хвилі, а потім оцінки згенерованих маршрутів розраховуються на основі нечіткого логічного висновку. Але хоча алгоритм, розроблений в цих дослідженнях, поліпшує якість обслуговування в програмно-визначених мережах, його практична імплементація може призвести до проблем з масштабуванням та рівномірним завантаженням каналів, оскільки використання нечіткої логіки зі збільшенням кількості переходів несе ризики збільшення гіпотетичності маршруту, який будується між хостами.

Ряд останніх публікацій займається пошуком нових рішень проблеми пошуку найкоротших шляхів. Так, у [36] пропонується ідея математичної пріоритезації потоків у вузли з кількома призначеннями. Надаючи перевагу маршрутам до кожного вузла призначення, запропонований алгоритм дозволяє трафіку швидше досягти адресатів. Тим не менше, пріоритетизація потоків хоч і може бути корисною, але цей підхід може призвести до нерівномірного розподілу ресурсів, тобто деякі потоки даних отримають менше ресурсів, ніж потрібно, що може знизити загальну продуктивність мережі.

Дещо схоже рішення запропоноване у [37] – алгоритм, розроблений у дослідженні, призначений для пошуку найкоротшого шляху в заданій мережі, що

задовольняє певні обмеження, під управлінням контролера *ONOS*. Але у цій статті автор не проаналізував масштабованість та відмовостійкість свого алгоритму, не визначена необхідність та можливість конвергенції на основі динамічних змін в мережі, як-от, зміни в структурі трафіку або відмови обладнання.

Є публікації, що пропонують маршрутизувати дані у програмно-визначеній мережі за вектором дистанції. Наприклад, у [38] застосована багатошляхова маршрутизація по критерію визначення маршруту, де під час визначення основного шляху до кінцевого хосту розраховуються також маршрути від проміжних ланок до нього, що сприяє рівномірнішому розподілу навантаження. Потім маршрутна інформація зберігається і для раніше сформованих шляхів заново не розраховується. Цей підхід є доволі перспективним та повністю реалізує концепцію *SDN*, але не дає виграшу у разі динамічних змін конфігурації мережі, що потребує переформатування налаштувань зі сторони контролера.

Пошук найкоротшого шляху у мобільних *SDN* мережах за допомогою алгоритму кластеризації запропонований у [39]. Зв'язки між кластерами та всередині кластерів використовуються для передачі даних з метою зниження часової складності маршрутизації, а за допомогою хвильового алгоритму формуються шляхи, що не перетинаються, між хостами. Отриманий результат усуває проблему відліку до нескінченності, що є головним недоліком алгоритму векторної маршрутизації по вектору дистанції, але потребує більше ресурсів у порівнянні з іншими рішеннями побудови програмно- визначених мереж.

У останні два роки були запропоновані найсучасніші рішення для вирішення проблем маршрутизації, реконфігурації та балансування завантаженості мереж *SDN*. Одним з таких є використання машинного навчання, огляд публікацій по якому виконали Фаезі та інші [40]. Згідно результатам дослідження, у більшості статей на цю тематику розглядалася реактивна модель для виявлення та ідентифікації, а архітектура проактивної моделі, яка використовується деякими розробниками для пом'якшення проблем з управлінням, є помітною прогалиною в останніх дослідницьких роботах, пов'язаних з машинним керуванням. Крім того, онлайн навчання майже не розглядалось в оглянутих публікаціях, хоча воно може

покращити продуктивність моделі, в тому числі при використанні реального середовища.

Автори у публікації [41] пропонують стратегію маршрутизації на основі метаевристичного алгоритму для досягнення оптимізації процесу маршрутизації. Ваги інерції в описаному алгоритмі постійно обробляються для адаптації до динамічних характеристик топології програмно-визначеної мережі, що покращує швидкість збіжності вихідного алгоритму. Крім того, вводяться кросовер і мутація генетичного алгоритму, щоб покращити здатність алгоритму знаходити підходящий шлях. Це одне з найперспективніших рішень на сьогоднішній день, що демонструє результати на мережах будь-якої розмірності, однак виставляє суттєві вимоги до апаратного забезпечення.

### **1.5. Аналіз відомих наукових підходів до підвищення відповідності SDN мереж якості обслуговування**

Для підвищення якості передачі трафіку та стабілізації мереж у наукових дослідженнях розглядаються способи покращення відповідності якості обслуговування (*Quality of Service, QoS*) власне мереж за рахунок оцінок метрик цієї якості за різними критеріями.

Контролери *SDN* мають повний контроль над мережами, і для підтримки належної *QoS* вони повинні бути розташовані певним чином. Проблеми, які можуть завадити досягненню необхідної якості обслуговування через використання контролерів, можуть включати в себе: безпеку мережі, управління мережею, ефективне використання ресурсів, програмованість мережі, управління мережевими послугами. Такі виклики у мережах *SDN* можна звести до мінімуму, підтримуючи надійність, масштабованість, узгодженість та балансування навантаження.

Хоча *SDN* є привабливою дослідницькою областю в сучасних комп'ютерних мережах, проблема у цій галузі полягає у тому, що для великої мережі розгортанням контролерів зазвичай важко керувати. Надійність і масштабованість

є ключовими проблемами для *QoS* у програмованій мережі. Саме тому проводяться різноманітні дослідження, щоб подолати ці виклики та відповісти на запитання, як-от: скільки контролерів потрібно включити в інфраструктуру *SDN* мережі, знайти місця для розгортання цих контролерів і також як вони будуть обмінюватися даними з підключеними пристроями. Відповідно, необхідно провести аналіз розміщення контролерів щодо метрик параметрів *QoS*.

Для покращення продуктивності мережі проектування та розміщення різних контролерів залишаються досліджуваними питаннями. Інші важливі аспекти, які слід враховувати для покращення продуктивності мережі, включають гнучкість, масштабованість, затримку, безпеку та узгодженість [22].

У великих мережах для балансування навантаження трафіку розгортаються кілька контролерів *SDN*. Коли кількість контролерів у мережах збільшується, концепція централізованого управління розпливається і втрачає свої переваги. Для досягнення хорошої *QoS* мережі потрібні кілька контролерів, оскільки їх спільна потужність більша, ніж у одного контролера. Однак централізоване управління стає складним через різні функції різних контролерів.

Наявність кількох контролерів полегшує масштабування програмно-визначених мереж. Однак масштабованість може вплинути на *QoS* через необхідність ефективного балансування навантаження між контролерами.

Основною проблемою в програмованих мережах із кількома контролерами є синхронізація інформації про стан мережі між ними. Ця проблема відома як проблема консенсусу. У випадку занадто складної реалізації обміну службовою інформацією між контролерами зростає затримка прийняття консенсусу. Відповідно, для проектування мережі з кількома контролерами необхідно спланувати процес досягнення узгодженості між ними.

З іншої сторони, використання лише одного контролера в мережі *SDN* має багато переваг, таких як управління, контроль і моніторинг усього мережевого середовища з одного вузла централізовано. Втім, централізація може бути як перевагою, так і недоліком та призводити до погіршення надійності та масштабованості в мережі. Зі збільшенням кількості трафіку та пристроїв ці

проблеми негативно впливають на продуктивність. Щоб зменшити затримку та покращити загальну продуктивність мережі, необхідно розмістити оптимальну кількість контролерів на відповідній відстані в мережах. Розрахунок необхідної кількості контролерів та вибір місць, де вони повинні бути встановлені в мережах є *NP*-складними задачами [42].

У розподіленому середовищі потрібна велика кількість контролерів, які безпосередньо впливають на *QoS* мережі *SDN*. Для вирішення цієї проблеми необхідний ефективний зв'язок між контролерами із відповідним спеціалізованим протоколом. Розробка цього протоколу є відкритою на сьогоднішній день задачею. Щоб сумісна робота різних контролерів була ефективною, необхідно уникнути їхнього перевантаження, швидко збалансувавши трафік у мережі.

У дослідженні [43] запропоновано механізм для прийняття рішень щодо розміщення контролерів, який визначає їхню необхідну кількість у фізичних *SDN* мережах. Цей механізм дозволив збільшити стійкість мережі *SDN*. У публікації [44] було запропоновано евристичний алгоритм для випадку, коли розв'язок оптимізаційної задачі запропонованої моделі неможливо отримати в практичний час, що дозволило знизити затримку передачі даних та вирішити проблему балансування навантаження для розміщення контролерів у *SDN*.

Паулсон та інші [45] запропонували підхід визначення ваги мінімально-максимального значення для вирішення компромісів, пов'язаних із проблемою оптимізації між контролерами. Функція вартості кількох обмежень мінімізується для однієї змінної. У статті [46] автори зменшили накладні витрати контролерів і оптимізували їх розташування для розміщення. Відповідно до передумови обмеження навантаження на локальну мережу, налаштування посилення для однієї спільної конкурентної гри вибирає учасників наступного раунду гри, і після застосування спільного підходу кілька разів загальна затримка мережі ефективно зменшується, а балансування навантаження призводить до покращеного розподілення трафіку та стабілізації мережі.

Автори у публікації [47] намагалися вирішити проблеми розміщення контролерів шляхом масштабування *SDN* мережі. Вони використали модель черги



очікування нескінченного буфера з кількома контролерами для аналізу показників продуктивності системи. Для ілюстрації запропонованої моделі використовуються як числові, так і змодельовані оцінки.

У статті [48] була запропонована схема адаптивного розміщення контролерів для архітектури систем *DT*. Враховуючи кореляцію між *DT* взаємодіючих і обчислювальних об'єктів, використане значення Шеплі теорії кооперативних ігор, яке справедливо кількісно визначає внески взаємодіючих обчислювальних об'єктів.

Рафід та інші [49] запропонували схему *SDVN*, яка на основі машинного навчання вибирає мінімальну тривалість зв'язку (*LD*) для зв'язку між вузлами, але вибирає шлях із максимальним часом шляху між джерелом і одержувачем. Двофазна схема вибору покращує продуктивність мережі з точки зору коефіцієнта доставки (майже 95%), наскрізної затримки (мінімальне значення 0,11 мс) і коефіцієнта витрат на маршрутизацію (максимальне значення становить 10%), що підтверджується результатами моделювання.

У публікації [50] автори запропонували кооперативну модель оптимізації *Salp Swarm (CMSSO)*, яка поєднує чотири алгоритми. Ці алгоритми співпрацюють, щоб вирішувати тести чисельної оптимізації з однією ціллю. Симуляція методу показала, що витрати на проектування *SDN* мережі знизились.

Одебайо та інші [51] розглянули проблему розподілу комутатора на контролер, яка враховує неоднорідність потужностей контролера. Вони пропонують два алгоритми на основі центральності сусідства для вирішення проблеми з метою мінімізації затримки перемикання на контролер. Також ними була представлена функція зваженої центральності, яка забезпечує справедливий розподіл навантаження між контролерами. Запропоновані алгоритми використовують вимірювання та евристики на основі центральності для визначення ідеального розподілу між комутаторами та контролерами, які враховують пропускну здатність відповідних вузлів контролера. Результати показують, що врахування неоднорідності потужностей контролера значно зменшує дисбаланс навантаження.

Автори у статті [52] використали власний підхід для зменшення кількості контролерів у великих *SDN* мережах. Вони розробили відмовостійкий механізм для вищезгаданого середовища. У цьому документі пропонується консенсусний протокол для підвищення відмовостійкості *SDN* з кількома контролерами.

Автори у публікації [53] запропонували гібридну інфраструктуру *SDN* для розподілу великих мереж між контролерами. Вони математично сформулювали проблему *EASON* як задачу цілочисельного програмування, довели її неполіноміальну часову складність і запропонували евристичний алгоритм під назвою *BonSec*. Експериментальні результати показують, що *BonSec* досягає рівня безпеки та економічності, які можна порівняти з оптимальним рішенням у невеликих топологіях. У той же час його можна масштабувати на великих топологіях.

У роботі [54] пропонується модель бінарного лінійного програмування для досягнення компромісу в глобальних мережах. Модель визначає тип, місце та мінімальну кількість необхідних контролерів, отримавши місце перемикачів, кількість процесорної потужності контролерів і вартість налаштування кожного контролера. Відповідно, це зменшує вартість налаштування, середню затримку між комутаторами та контролерами, а також середню затримку між контролерами порівняно з існуючими методами. Результати дослідження показують, що модель має найнижчу вартість налаштування та мінімальну середню затримку контрольних пакетів (від комутатора до контролера та контролера до контролера) при розміщенні контролерів порівняно з попередніми методами.

Кумар [55] реалізував модифікований алгоритм Брезенхема та алгоритм Дейкстри, щоб знайти підходящий шлях за значно скорочений час обчислення. Крім того, використовуючи переваги детермінованої групи супутників, був запропонований алгоритм передачі правила потоку та алгоритм моніторингу топології.

У публікації [56] пропонується властивість під назвою суфіксна причинно-наслідкова узгодженість (*SCC*). Вона оцінюється у *Open vSwitch* і комутаторах *P4* у поєднанні з контролерами *Ryu* та *P4Runtime*. Результати показують, що *SCC*

забезпечує більшу ефективність, ніж конкуруючі альтернативи послідовного оновлення, водночас пропонуючи послідовність, яка є достатньо сильною, щоб забезпечити високорівневі властивості маршрутизації.

У публікації [57] автор описує стратегію відновлення шляху керування, коли контролер виходить з ладу в площині керування локальною *SDN* мережею, виявляє відповідну математичну модель, пропонує відповідний алгоритм відновлення шляху та демонструє відповідні експерименти, що доводять досягнення мережею необхідного рівня якості обслуговування.

У статті [58] автори розробили власну архітектуру, у якій *SuperController* зв'язується з *RegularControllers*, розділеними на кластери. Щоб гарантувати, що всі контролери в мережі мають повну інформацію про всю мережу та повідомляють іншим контролерам, що в мережі відбувається певне оновлення, використовується спільне сховище даних. Отже, кожен кластер має власне сховище даних для спільного використання. Крім того, існує глобальне сховище даних спільного доступу для всієї мережі, до якого можуть отримати доступ усі *SuperController* і звичайні контролери в одному кластері.

Тівана та Сінгх [59] пропонують техніку адаптивної оптимізації навантаження для мультимедійних програм із використанням кількох контролерів у *SDN*. Ключем до цього методу є адаптивний механізм оптимізації навантаження, який розумно розподіляє мережеві ресурси відповідно до поточних моделей трафіку та потреб додатків. Метод виявляє перевантаження мережі та проактивно перерозподіляє навантаження між сегментами мережі за допомогою методів машинного навчання та оптимізації. Для мультимедійних програм це гарантує оптимальне використання ресурсів, зменшує затримку та покращує якість обслуговування мережі (*QoS*).

Хоа та інші [60] запропонували контейнерну архітектуру, здатну швидко масштабувати вгору та вниз на основі навантаження трафіку для площини керування *SDN*. За допомогою нового адаптивного до трафіку алгоритму результати показують, що запропонована система здатна забезпечити

продуктивність із високим числом вхідних запитів нового потоку та зменшити недостатньо використовувані контролери для ефективного використання ресурсів.

У публікації [61] була представлена нова техніка балансування за унікальним методом оцінки навантаження, який усереднює навантаження на контролери *SDN*, таким чином сприяючи ефективному розподілу трафіку. Продуктивність запропонованої методики була оцінена на основі ступеня балансування навантаження, часу відгуку мережі та вартості міграції.

Нарешті, найоригінальніший підхід до покращення *QoS* мережі *SDN* був представлений у [62]. Був описаний метод оптимізації мурашиної колонії для перемаршрутизації з урахуванням балансування навантаження в мережах *SDN*. Ця техніка, яка називається *DPLBAnt*, сформульована як проблема найкоротшого шляху в *SDN*, і може зменшити високе навантаження на комутатор контролера. Представлена техніка *roposed* спочатку виявляє потоки даних за допомогою пари класифікаторів як на контролері *SDN*, так і на комутаторах. Більшість кандидатів відсіюються на комутаторах, що забезпечує точне та ефективне виявлення. Потім *DPLBAnt* отримує глобальний стан *SDN*, з якого витягуються найбільш підходящі шляхи для перевантажених посилянь, і потоки даних перенаправляються відповідно. Продуктивність запропонованого *DPLBAnt* була змодельована, результати свідчать про кращу продуктивність комп'ютерної мережі порівняно з існуючими методами.

Огляд описаних в цьому пункті наукових досліджень показує, що у всіх роботах автори вирішують проблему відповідності мереж *SDN* якості обслуговування доволі нетривіальними методами математики або програмування, використовуючи складні обчислювальні алгоритми. Крім того, більшість оглянутих рішень виставляє суттєві вимоги до апаратного забезпечення – хмарні сервери, кластери контейнерів, тощо. З огляду на це, постає проблема розробити такий спосіб підвищення відповідності мережі *SDN* якості обслуговування, який не потребує таких значних апаратних вимог та вирішує проблему побудови шляхів відносно простими для програмної реалізації алгоритмами.

## Висновки до розділу 1

Аналіз можливостей технології *SDN* та її порівняння з традиційними методами побудови мереж показують, що на сьогодні актуальною є необхідність розробки методів побудови каналів зв'язку, що дозволять запобігти перевантаженню зв'язків між вузлами мережі та знизити втрати даних під час сеансу передачі. Також проблемою, що потребує вирішення, є розробка методів динамічної реконфігурації мереж задля забезпечення безперебійного сеансу передачі трафіку, адаптації до змін топології, приєднання нових сегментів, відмов окремих вузлів.

В ході аналізу технології віртуальних мереж було досліджено їхні можливості впроваджувати складні алгоритми управління трафіком та багатофункціональні мережеві служби без необхідності змін на апаратному рівні. Ці можливості є важливим доповненням до проектування програмно-визначених мереж у контексті безпеки. Віртуальні мережі можуть вирішити ряд актуальних проблем у мобільних інфраструктурах великої розмірності, які мають тенденцію до постійного зростання: реконфігурація мережі у разі зростання кількості користувачів, впровадження нових протоколів передачі даних.

Використання віртуальних мереж в поєднанні з технологією *SDN* зміцнить адаптивність, масштабованість та продуктивність мережі. На основі оглянутого матеріалу була поставлена задача розробки методу динамічної реконфігурації віртуальної *SDN*-мережі, який дозволить підтримувати безперебійний сеанс передачі трафіку у разі відмови основних каналів зв'язку.

Аналіз літературних джерел показав, що хоча традиційні алгоритми пошуку шляхів забезпечували прийнятну швидкість передачі даних у розподілених мережах, програмно-визначені мережі потребують нових підходів до процесу побудови шляхів. Нинішні обчислювальні можливості та централізація управління дозволяють застосувати більш гнучкі підходи для збільшення ефективності побудови каналів зв'язку. Одним з таких підходів є забезпечення якості обслуговування мережі *SDN* задля гнучкості її використання в умовах підвищених

навантажень та зростання кількості користувачів. Втім, аналіз існуючих рішень у наукових публікаціях показав, що з усіх запропонованих методів немає такого, який би був достатньо ефективним при збереженні відносно невеликих вимог до апаратного забезпечення. Тому була поставлена задача розробити відповідний метод, де побудова віртуальних каналів зв'язку буде відбуватися з урахуванням якості обслуговування.

## РОЗДІЛ 2

### ПОБУДОВА КАНАЛІВ ЗВ'ЯЗКУ ВІРТУАЛЬНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN ІЗ ЗАДАНОЮ ЯКІСТЮ ОБСЛУГОВУВАННЯ

#### 2.1. Математичне обґрунтування проблеми побудови каналів зв'язку з урахуванням параметрів якості обслуговування

Топологію будь-якої комп'ютерної мережі можна представити як орієнтований граф  $G(V, E)$ , де  $V$  означає всі вузли мережі, а  $E$  означає всі з'єднання між ними. Кожне з'єднання  $e_{(i,j)} \in E$  має власні властивості, такі як доступна смуга пропускання  $\lambda_e$ , затримка  $\mu_e$ , та втрата пакетів  $\rho_e$ .

Коли робиться запит на маршрутизацію, статус і показники мережі оновлюються, щоб підвищити точність даних про доступні ресурси та визначити шлях, який задовольняє параметри якості послуг [63]. Завдання пошуку найкоротшого шляху полягає в тому, щоб знайти маршрут  $P$  від вузла джерела  $s$  до вузла призначення  $d$ , який задовольняє різні значення параметрів мережі та при цьому дозволяє досягти менших часових витрат на зв'язок між вузлами, ніж альтернативні маршрути. Для шляху  $P$  визначимо функцію його побудови.

Основні метрики якості обслуговування, які враховуються при побудові маршрутів, є наступними.

Метрика хопів (*hops*) позначає кількість переходів між вузлами  $s$  і  $d$ . У цьому випадку вартість кожного переходу (і відповідно, вага кожного ребра графу топологічного малюнку) дорівнює 1.

$$h(P) = \sum_{e \in P} e, \quad (2.1)$$

де  $h(P)$  – кількість переходів (хопів) у шляху.

Метрика смуги пропускання (*bandwidth*) позначає мінімальну пропускну здатність між вузлами  $s$  і  $d$ , а метрика вартості кожного каналу встановлюється обернено пропорційною пропускну здатності цього каналу.

$$\lambda(P) = \min_{e \in P} \left\{ \frac{1}{\lambda_e} \right\}, \quad (2.2)$$

де  $\lambda(P)$  – мінімальна з доступних смуг пропускання для переходів на шляху.

Метрика затримки (*delay*) шляху є сумою усіх затримок, що виникають при проходженні даних на кожному переході між ланками.

$$\mu(P) = \sum_{e \in P} e_\mu, \quad (2.3)$$

де  $\mu(P)$  – загальна сумарна затримка шляху.

Метрика втрати пакетів (*packet loss*) позначає відсоток втрачених пакетів у порівнянні з загальною кількістю надісланих пакетів (відповідно, нульове значення метрики означає відсутність втрат пакетів):

$$\rho(P) = 1 - \prod_{e \in P} (1 - e_\rho), \quad (2.4)$$

де  $\rho(P)$  – відсоток втрачених пакетів відносно кількості надісланих пакетів.

У існуючих рішеннях при виконанні задач побудови віртуальних мереж та маршрутизації завжди враховуються лише окремі метрики якості обслуговування, але не усі разом. Це зумовлено навантаженням на низькопродуктивну апаратну частину маршрутизуючих пристроїв, яку розробники цієї апаратури зазвичай не розраховують під великі обчислювальні навантаження. Наприклад, протокол *RIP* враховує лише кількість переходів між вузлами при побудові свого дерева маршрутизації. Протокол *OSPF* виконує пошук найкоротших шляхів по кількості переходів та з урахуванням статусу каналу зв'язку (лише доступною пропускнуою можливістю), що обмежує можливості побудови якомога ефективнішого зв'язку у мережі [64].

Функція затримки є адитивною (тобто значення загальної затримки завжди дорівнює сумі затримок на окремих каналах, в незалежності від їхнього набору у шляху), тоді як функція пропускнуї здатності є угнутою (має від'ємну другу похідну). Функція втрати пакетів є мультиплікативною (значення функції для добутку параметрів дорівнює добутку значень функції від кожного з цих параметрів), але її можна перетворити на адитивну функцію, взявши логарифм



відношення. Саме за допомогою цього параметра і є можливим забезпечити мережевий *QoS* [65].

Щоб оцінити продуктивність шляху, розглянемо пропускну здатність як міру продуктивності, яка характеризує успішну доставку даних  $E_{s,t}$  через канал зв'язку. У даному випадку це шлях, де  $E_{s,t}$  є отриманими даними  $d_{rx}$  від джерела  $s_{tx}$ , за винятком повторно переданих даних  $r_{data}$  протягом усього часу спостереження  $T_t$  [66]:

$$t_h = \frac{E_{s,t}}{T_t}, \quad (2.5)$$

де, відповідно:

$$E_{s,t} = \frac{s_{tx} - (d_{rx} - r_{data})}{s_{tx}}. \quad (2.6)$$

Алгоритм найкоротшого шляху має визначати маршрут із найнижчою вартістю з'єднання, визначеною значенням, обернено пропорційним доступній пропускній здатності кожного з'єднання. Зазвичай для такого типу пошуку використовується алгоритм Дейкстри. Оскільки для побудови маршрутів було обрано для урахування чотири різні метрики, то варіанти пошуку найкоротшого шляху мають бути об'єднані у єдину вихідну матрицю та обраховані за допомогою алгоритму Дейкстри. Необхідно розглянути показники шляху з найменшою кількістю переходів, вартістю пропускнуї здатності, затримкою та втратою пакетів [67].

Шлях мінімальних переходів позначає найкоротший шлях із найменшою кількістю переходів між вихідним і кінцевим вузлом.

$$H_{ij} = \min(h(P)), \quad (2.7)$$

де  $i$  – початкова вершина, з якої будується шлях;  $j$  – кінцева вершина шляху.

Шлях мінімальної вартості смуги пропускання позначає найменшу доступність пропускнуї здатності з-поміж доступних переходів.

$$B_{ij} = \min(\lambda(P)) \quad (2.8)$$

Шлях мінімальної затримки позначає найменшу з сумарних затримок доступних маршрутів.

$$D_{ij} = \min(\mu(P)) \quad (2.9)$$

Шлях мінімальної втрати пакетів позначає найменший відсоток втрачених пакетів при передачі даних з-поміж доступних маршрутів.

$$L_{ij} = \min(\rho(P)) \quad (2.10)$$

У роботі ставиться мета будувати найкоротші шляхи між пристроями у комп'ютерній мережі з використанням різних метрик *QoS*. Змінюючи мінімальну вартість пересилань для метрики, за якою необхідно оцінити побудову найкоротшого маршруту, є можливість динамічно вибрати шлях для даних між вузлом-джерелом та вузлом-адресатом.

## **2.2. Метод побудови каналів зв'язку на основі інтегрального критерію з урахуванням метрик якості обслуговування**

### **2.2.1. Обчислення інтегрального критерію побудови каналу зв'язку**

Метод, що пропонується, заснований на застосуванні класичного алгоритма Дейкстри. Він базується на вивченні набору сусідніх вузлів і обчисленнях для визначення потенційного найкоротшого шляху між джерелом *s* і пунктом призначення *d*. Власне, він обчислює найкоротші шляхи до всіх пунктів призначення. Потім ця інформація використовується *SDN*-контролером для маршрутизації по топологічному графу [68].

Щоб провести обчислення шляхів між вершинами, алгоритм Дейкстри потребує вагову матрицю, на основі якої він порівнюватиме переходи (ваги ребер на графі) та комбінаторно визначатиме найкоротший з них. Як уже зазначалося, у традиційному підході зазвичай враховується лише одна (рідше – дві) метрики переходів. У даній роботі пропонується враховувати чотири – кількість переходів між початковою та кінцевою вершинами, смуга пропускання, затримка та втрата пакетів.

На основі вагової матриці алгоритм формує маршрут проходження пакетів. З урахуванням обраних параметрів якості обслуговування, вибір маршруту буде

більше відповідати потребам користувача.

Нехай у процесі передачі трафік повинен досягти вершини-адресата з гарантією та у найкоротші терміни. Враховуючи роботу алгоритму Дейкстри, трафік пройде по єдиному найкоротшому (але не обов'язково найбезпечнішому) маршруту. У процесі передачі трафіку він має пройти кожен проміжний пристрій у мережі відповідно до топологічного малюнку, рухаючись таким чином по шляху, який відповідає заданим вимогам.

Для розрахунку шляху за допомогою алгоритму Дейкстри потрібно обчислити сумарну вартість проходження трафіку між двома вузлами  $i$  та  $j$ , та обрати шлях з найменшою вартістю до вузла  $j$ , де  $j \neq i$ .

Це можна представити у вигляді двох параметрів:

- $d_{i,j}$  – вартість зв'язків між вершинами  $i$  та  $j$ ;
- $D_{i,j}$  – найкоротший шлях, яким проходитиме трафік між вершинами  $i$  та  $j$ .

На початку своєї роботи алгоритм Дейкстри позначає вершини заданого топологічного графа. Він використовує два списки для позначення мінімальної вартості вузлів: один містить обчислені вузли (постійний список  $S$ ), а інший – вузли, яких ще немає (попередній список  $S'$ ). Протягом кожної ітерації алгоритм обчислює найкоротший шлях від сусіднього вузла  $k$  до вузла  $i$  з найменшою вартістю до вузла  $i$ . Список  $S$  збільшується шляхом додавання обчислених вузлів. Список  $S'$  при цьому зменшується шляхом видалення вузлів, що були додані до  $S$  [69].

Алгоритм починається з призначення постійної мітки 0 початковій вершині  $s$  та тимчасової мітки нескінченності решті  $(n-1)$  вершин. Відтоді на кожній ітерації кожна наступна вершина встановлює постійну мітку відповідно до певних правил.

Кожна вершина  $j$ , яка ще не позначена постійно, отримує нову тимчасову мітку, значення якої задається наступним чином:

$$D_{ij} = \min(D_{ij}, (d_i + w_{ij})), \quad (2.11)$$

де  $D_{ij}$  – мітка вершини  $j$  (у правій частині рівняння – «стара» мітка у списку  $S'$ ,

визначена як мінімальний шлях між вершинами  $i$  та  $j$  на минулій перевірці);  $d_i$  – «стара» мітка вершини  $i$  у списку  $S$ ;  $w_{ij}$  – вага ребра (пряма відстань) між вершинами  $i$  та  $j$ . Якщо  $i$  та  $j$  не з'єднані ребром, тоді  $w_{ij} = \infty$ ;  $i$  – остання вершина, постійно позначена на попередній ітерації алгоритму.

Після того як найменше значення серед усіх тимчасових міток знайдено, воно стає постійною міткою відповідної вершини (видаляється зі списку  $S'$  та переходить у список  $S$ ). У разі, якщо у кількох найкоротших шляхів однакова довжина, з них для постійного маркування вибирається один випадковим чином. Робота алгоритму продовжується, поки вершина призначення  $t$  не отримає постійну мітку. Перша вершина, яку потрібно постійно позначити, знаходиться на найменшій відстані від  $s$ . Друга вершина, яка отримує постійну мітку з решти  $(n-1)$  вершин, є вершиною, найближчою до  $s$ . З решти  $(n-2)$  вершин, наступною, яка буде постійно позначена, є друга найближча вершина до  $s$ , і так далі. Постійною міткою кожної вершини є найкоротша відстань цієї вершини від  $s$ .

Як зазначалося вище, для побудови найкоротших шляхів у роботі пропонується обрати чотири критерії вибору маршруту на основі параметрів обслуговування каналів зв'язку, що будуть скомбіновані у єдиний інтегральний критерій. Але алгоритм Дейкстри може одночасно працювати лише з одним набором даних з інформацією про ваги ребер графа. Щоби врахувати усі параметри якості обслуговування одночасно, потрібно вирішити наступні проблеми:

1. Об'єднати дані про ваги ребер за відповідними параметрами якості обслуговування каналів зв'язку;
2. Визначити (зі сторони користувача) пріоритетність окремих метрик якості обслуговування над іншими у якості критерія для обрання остаточного шляху.
3. Врахувати можливість розділення мережі на підмережі (сегменти) з однією чи кількома вузловими вершинами та необхідність передачі трафіку з одного сегменту в інший.

Для вирішення першої проблеми формуються відповідно чотири умовні матриці (див. буквенні позначення у формулах 2.7–2.10): матриця переходів  $H_{ij}$ ,

матриця смуги пропускання  $B_{ij}$ , матриця затримки  $D_{ij}$  і матриця втрати пакетів  $L_{ij}$ . Кожна умовна матриця використовує за вагу ребра порівняльні дані відповідної метрики для зв'язків між пристроями мережі – так, наприклад для порівняння втрати пакетів контролер *SDN* розсилатиме по мережі тестовий пакет даних для визначення відсотку втрат, і виставить значення вагів ребер пропорційно отриманим результатам для різних зв'язків між комутаторами. Потім умовні матриці потрібно просумувати для формування єдиної вагової матриці інтегрального критерію обрання шляху  $K_{ij}$ , з якою і працюватиме алгоритм Дейкстри.

Виходячи з вимог для адміністрування, пропонується визначення пріоритетів параметрів обслуговування цілими числами, що стануть значеннями величин, що пропонуються – коефіцієнтів пріоритезації. Наприклад, якщо врахування доступності смуги пропускання має більшу важливість для обміну даними у мережі, ніж кількість необхідних переходів між комутаторами, то коефіцієнт пріоритезації для матриці  $B_{ij}$  буде встановлений вищим за коефіцієнт матриці  $H_{ij}$  на величину, що обирається вимогами до мережі.

Можливе розділення мережі на сегменти передбачає проведення обчислень вагів ребер графу підмережі окремо в кожному сегменті. Для обміну даними між пристроями в межах одного сегмента додаткові дії, окрім вже описаних, не потребуються. У випадку, якщо дані мають пройти з одного сегменту в інший через вузлові вершини, сегменти варто вважати єдиною сутністю. Таким чином, для  $x$  сегментів їхні матриці інтегрального критерію  $K_{ij}$  будуть об'єднані між собою через сумування, але коефіцієнти пріоритезації у кожному сегменті можуть бути визначені окремо та враховані на етапі сумування умовних матриць сегменту.

Загальне математичне представлення формування вагової матриці на основі інтегрального критерію пошуку шляху  $K_{ij}$  для роботи оптимізованого алгоритму Дейкстри для пошуку найбільш збалансованого шляху між пристроями мережі має наступний вигляд [70]:

$$K_{ij} = \sum_{x \in N^*} (a \cdot H_{ij} + b \cdot B_{ij} + c \cdot D_{ij} + d \cdot L_{ij})_x, \quad (2.12)$$

де  $a$  – коефіцієнт пріоритезації переходів;  $b$  – коефіцієнт пріоритезації смуги пропускання;  $c$  – коефіцієнт пріоритезації затримки;  $d$  – коефіцієнт пріоритезації втрат пакетів;  $x$  – порядковий номер сегменту (підмережі).

Рішення не обмежується застосуванням лише чотирьох запропонованих метрик якості обслуговування та може бути розширене до необхідної кількості інших метрик з відповідними коефіцієнтами. В узагальненому вигляді формула обчислення інтегрального критерію має вигляд:

$$K_{ij} = \sum_{x \in N^*} \left( \sum_{y=1}^n k_y \cdot M_{ij_y} \right)_x, \quad (2.13)$$

де  $k$  – коефіцієнт пріоритезації метрики;  $M_{ij}$  – числове вираження метрики.

### **2.2.2. Математичне обґрунтування зменшення часової складності побудови віртуальних каналів зв'язку**

Важливим аспектом оцінки ефективності методу побудови каналів зв'язку є часова складність, тобто математичне представлення очікуваних часових затрат на побудову кожного каналу. У випадку алгоритму Дейкстри, після аналізу і порівняння з'єднань за певною умовою інформація формується у вигляді матриці вагів зв'язків між вершинами. Такими умовами можуть бути не лише певні евристичні правила за типом критеріїв якості обслуговування, а й цілком конкретні вимоги, як-от, у процесі логістики та дистрибуції необхідно пройти через низку необхідних вузлових вершин на топологічному графі, або ж у процесі вибору шляху необхідно кілька разів пройти через один і той самий вузол. Вимога може бути поставлена таким чином, що вибір цих вузлів може не мати нічого спільного з іншими метриками. Виходячи з цього, на загальну вагу зв'язків між вузлами впливатимуть не лише обраховані критерії порівняння маршрутів, а й обмеженість розгалуженості шляхів через необхідність повертатися у певні вершини.

Часова складність створення одного віртуального каналу від вихідної точки до кінцевого вузла у топологічному графі алгоритмом Дейкстри становить [71]:

$$O(N^2), \quad (2.14)$$

де  $N$  – кількість вузлів початкової фізичної мережі, в рамках якої формується канал.

Відповідно, часова складність створення  $k$  віртуальних каналів цим методом дорівнюватиме:

$$O(kN^2) \quad (2.15)$$

Зазвичай, одному віртуальному каналу може відповідати декілька фізичних, які, як правило, не перетинаються. Це допомагає підвищити ефективність процедури конструювання трафіку, забезпечити задані параметри  $QoS$  при зміні параметрів мережі, зокрема при зміні окремих метрик каналів передачі інформації, шляхом заміни одного фізичного каналу іншим в рамках одного віртуального каналу.

В загальному випадку, часова складність створення одного віртуального каналу, який складається з  $q$  фізичних каналів, складає:

$$O(qN^2) \quad (2.16)$$

При формуванні віртуального каналу, який складається з множини шляхів, що не перетинаються, а вершини, що входять у попередні шляхи, не враховуються, часова складність алгоритму складає [72]:

$$O\left(N^2 + \sum_{i=2}^k \left(N - \sum_{j=1}^{i-1} N_{L_j}\right)^2\right), \quad (2.17)$$

де  $N$  – кількість вузлів в мережі;  $k$  – кількість шляхів, що не перетинаються;  $N_{L_j}$  – кількість проміжних вершин шляху  $L_j$ .

Нехай представлений топологічний граф комп'ютерної мережі з певною зв'язаністю між 21 вершинами. Необхідно побудувати  $k$  маршрутів між першою та останньою вершинами за допомогою комбінаторного алгоритму таким чином, щоб вони не мали спільних переходів та вершин. Ваги ребер графа для даної задачі не прописані безпосередньо, але вважається, що перехід має якнайбільшу ціну в залежності від своєї довжини, що на практиці часто обумовлено географічними

особливостями. Таким чином, стоїть завдання побудови шляхів виключно відносно позицій на графі (рис. 2.1).

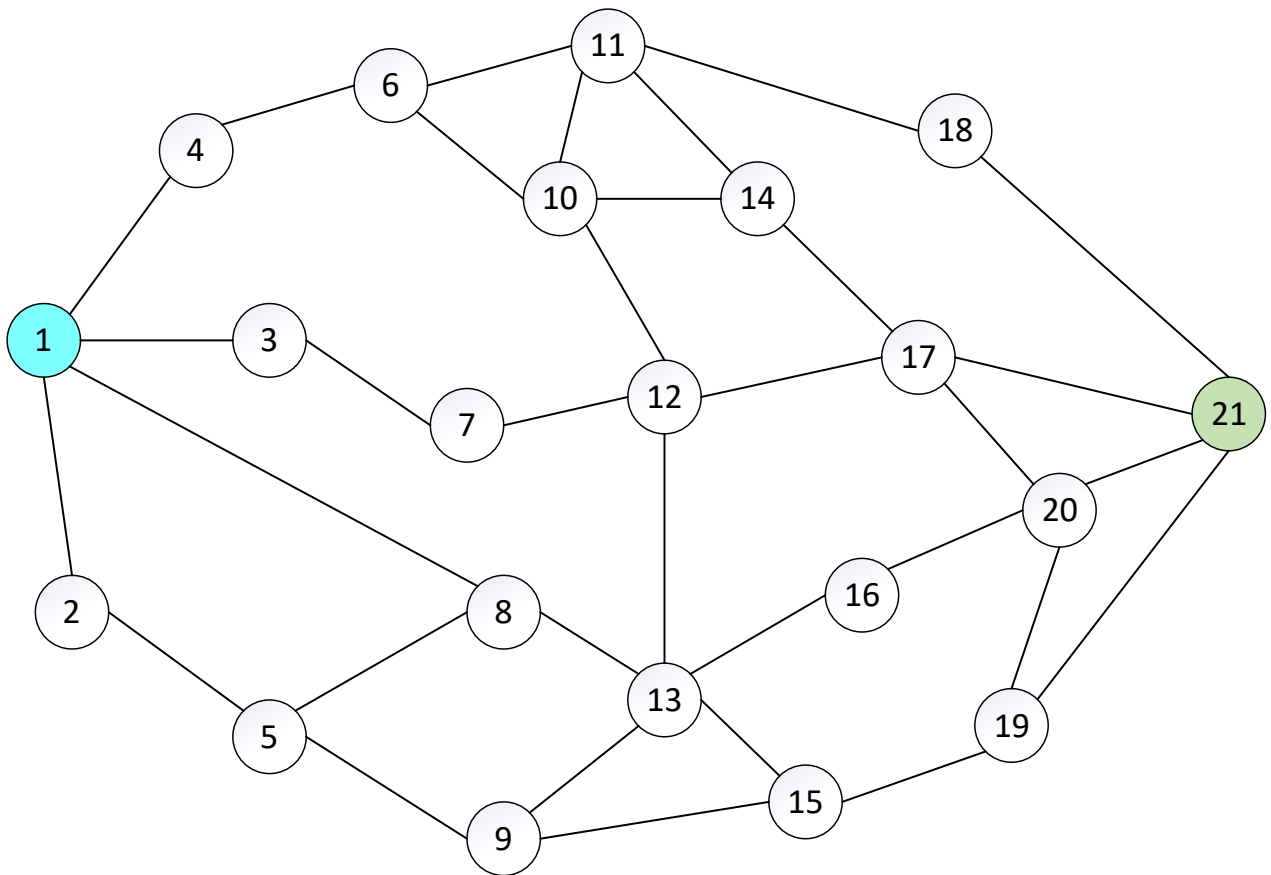


Рис. 2.1. Топологічний граф комп'ютерної мережі

Мінімальна роздільна множина цього графу дорівнює трьом, що можна довести з використанням комбінаторного алгоритму Дейкстри. Розглянемо його роботу покроково.

На першому кроці обирається найкоротший прямий шлях від початкового вузла 1 до вузла-адресата 21. Часова складність його пошуку складає, відповідно до базової складності алгоритму Дейкстри, квадрат кількості всіх вершин графу (рис. 2.2):

$$L_1: 1 \rightarrow 3 \rightarrow 7 \rightarrow 12 \rightarrow 17 \rightarrow 21, \quad (2.18)$$

$$t(L_1) = O(N^2) = 21^2 = 441. \quad (2.19)$$



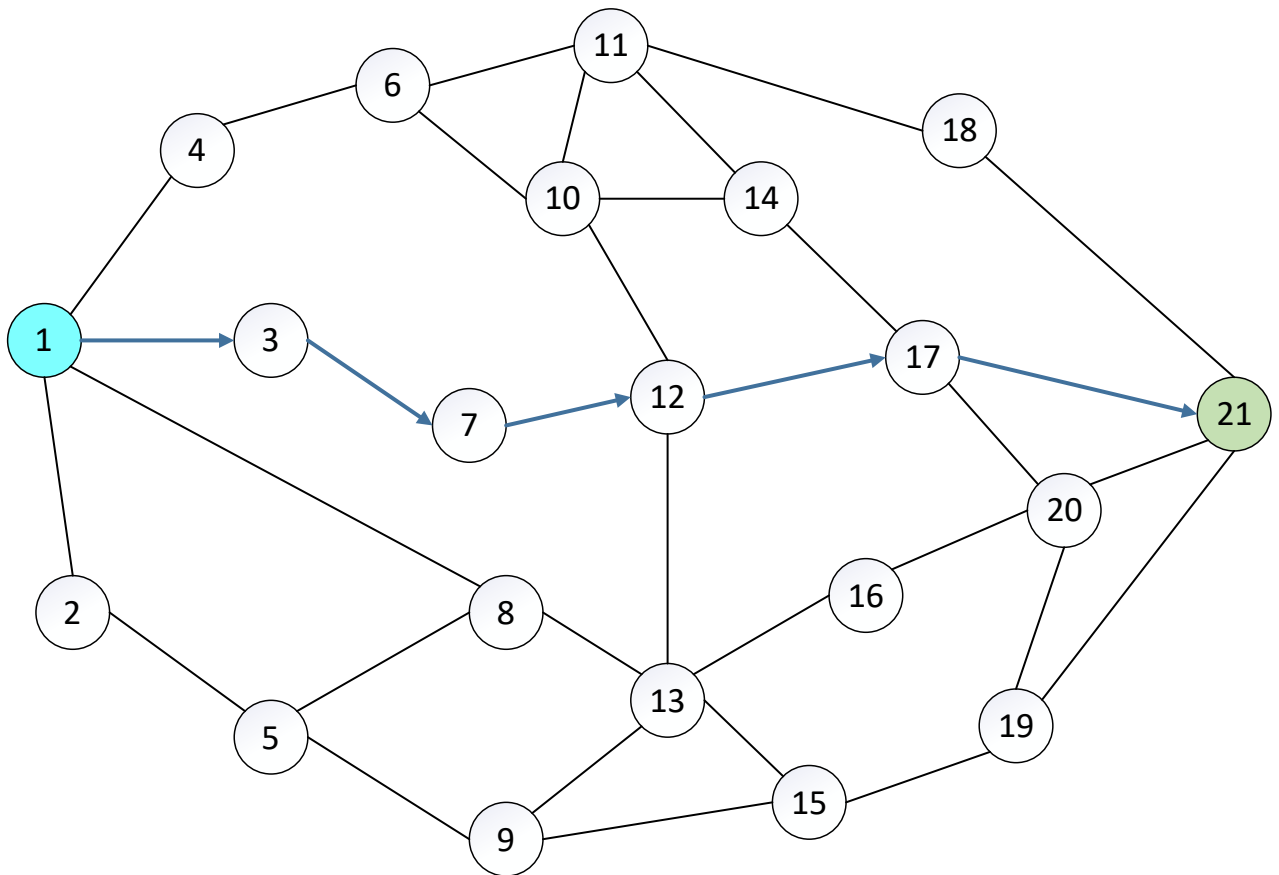


Рис. 2.2. Побудова першого віртуального каналу зв'язку між вершинами 1 і

21

На другому кроці формується шлях через чотири вершини у верхній частині графу. В рамках цілого графу часова складність його пошуку з урахуванням часу, витраченого на перший шлях, складатиме суму складності пошуку першого шляху та квадрат кількості вершин без урахування тих, через які вже проходить перший шлях.

На рисунку 2.3 позначено, окрім знайденого другого шляху, також набір вершин і переходів між ними, які не можуть бути задіяні у другому шляху через те, що шляхи не мають перетинатися між собою:

$$L_2: 1 \rightarrow 4 \rightarrow 6 \rightarrow 11 \rightarrow 18 \rightarrow 21, \quad (2.20)$$

$$t(L_2) = O(N^2 + (N - N_{L_1})^2) = 21^2 + (21 - 4)^2 = 730. \quad (2.21)$$

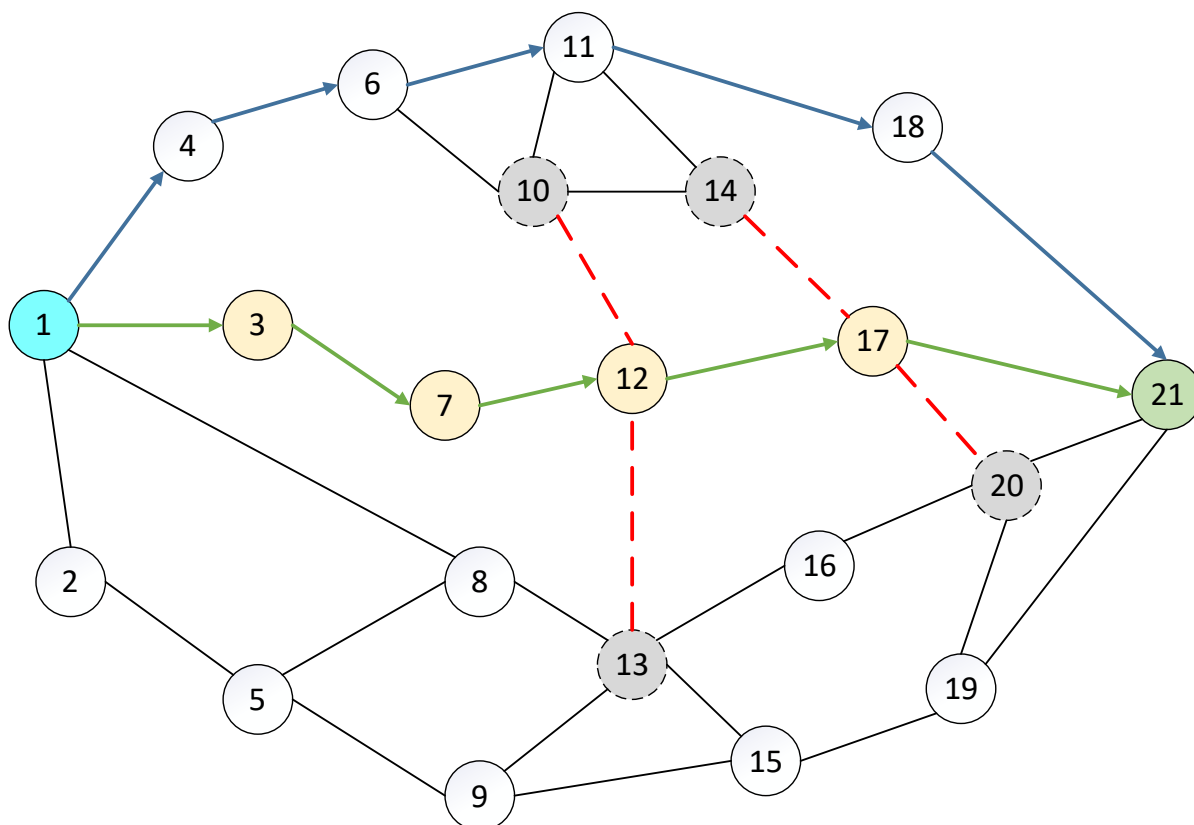


Рис. 2.3. Побудова другого віртуального каналу зв'язку між вершинами 1 і

21

На наступному кроці формується третій шлях, у нижній частині графу. Тут наявні два потенційні маршрути, що не перетинаються: той, що складається з чотирьох вершин (через вершини 8-20), та той, що включає в себе уже 5 проміжних вершин (через 2-19). Не дивлячись на те, що перший шлях є коротшим як за кількістю переходів, так і географічно (за умовами задачі, довжина зв'язку характеризує його вагу), він не може бути обраний на даній ітерації через те, що включає в себе вершини, поєднані з тими, що уже задіяні у раніше сформованих шляхах.

Тож обирається шлях через вершини 2 і 19 з шістьма переходами. До часової складності пошуку цього шляху разом з двома попередніми додається квадрат кількості вершин, які залишались на повному графі без урахування тих, що вже були задіяні у минулих шляхах. Схематичне представлення побудови шляху з «недоступними» переходами та вершинами через їхню «зайнятість» у минулих шляхах позначено на рис. 2.4:

$$L_3: 1 \rightarrow 2 \rightarrow 5 \rightarrow 9 \rightarrow 15 \rightarrow 19 \rightarrow 21, \quad (2.22)$$

$$t(L_3) = O(N^2 + (N - N_{L_1})^2 + (N - N_{L_2})^2) = 730 + (21 - 8)^2 = 899 \quad (2.23)$$

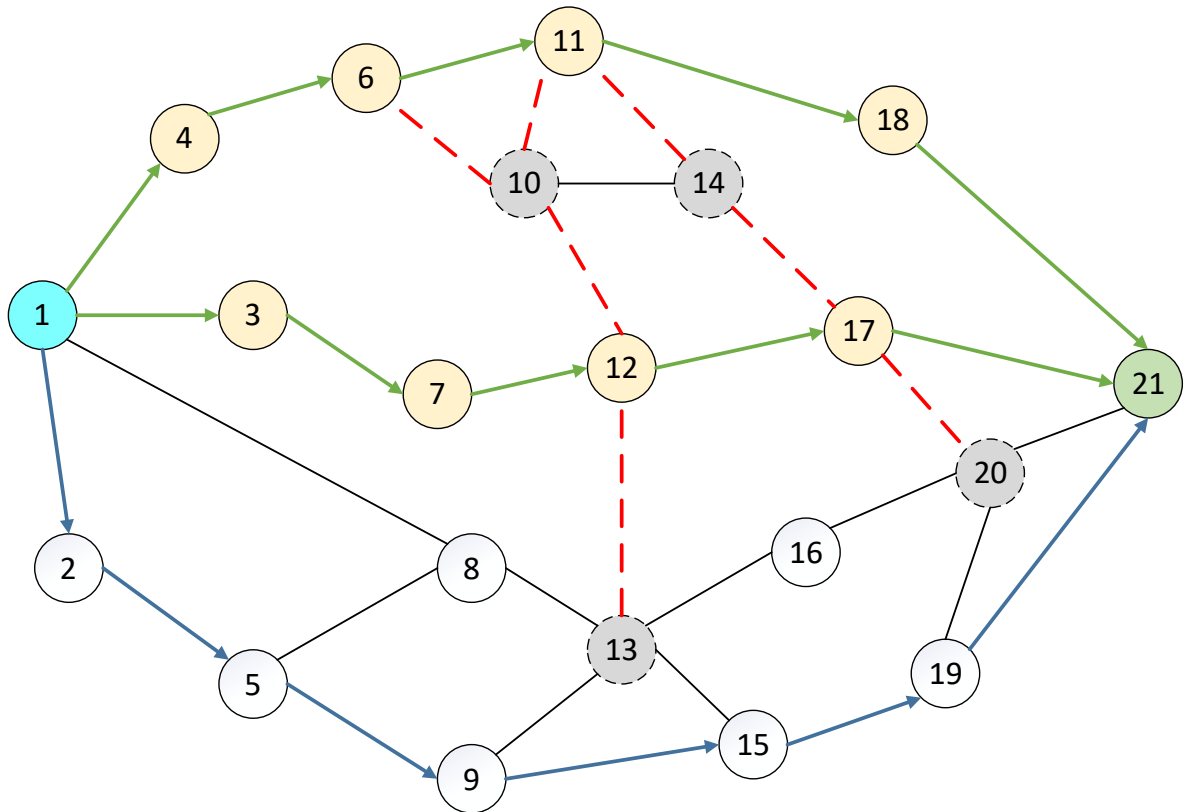


Рис. 2.4. Побудова третього віртуального каналу зв'язку між вершинами 1 і

21

Четвертий шлях формується також у нижній частині графа з чотирьох проміжних вершин. На даному етапі вже не залишилось вершин, що не зв'язані з тими, що були задіяні у минулих шляхах, тому відбувається спроба пошуку проміжних вершин, що мають зв'язки не лише з тими, що задіяні раніше, але й з незадіяними. Це дає змогу обрати відкинутий на попередній ітерації маршрут через вершини 2 і 19. Підтверджується правильність формульного представлення часової складності пошуку шляху згідно (2.16). Схематично шлях позначений на рис. 2.5:

$$L_4: 1 \rightarrow 8 \rightarrow 13 \rightarrow 16 \rightarrow 20 \rightarrow 21, \quad (2.24)$$

$$t(L_4) = O\left(N^2 + \sum_{i=2}^4 \left(N - \sum_{j=1}^3 N_{L_j}\right)^2\right) = 899 + (21 - 13)^2 = 963. \quad (2.25)$$



Виконаємо переоцінку часової складності пошуку шляхів, що не перетинаються, для графу, що розглядається у даному пункті, з урахуванням його динамічного розбиття на підграфи (рис. 2.6).

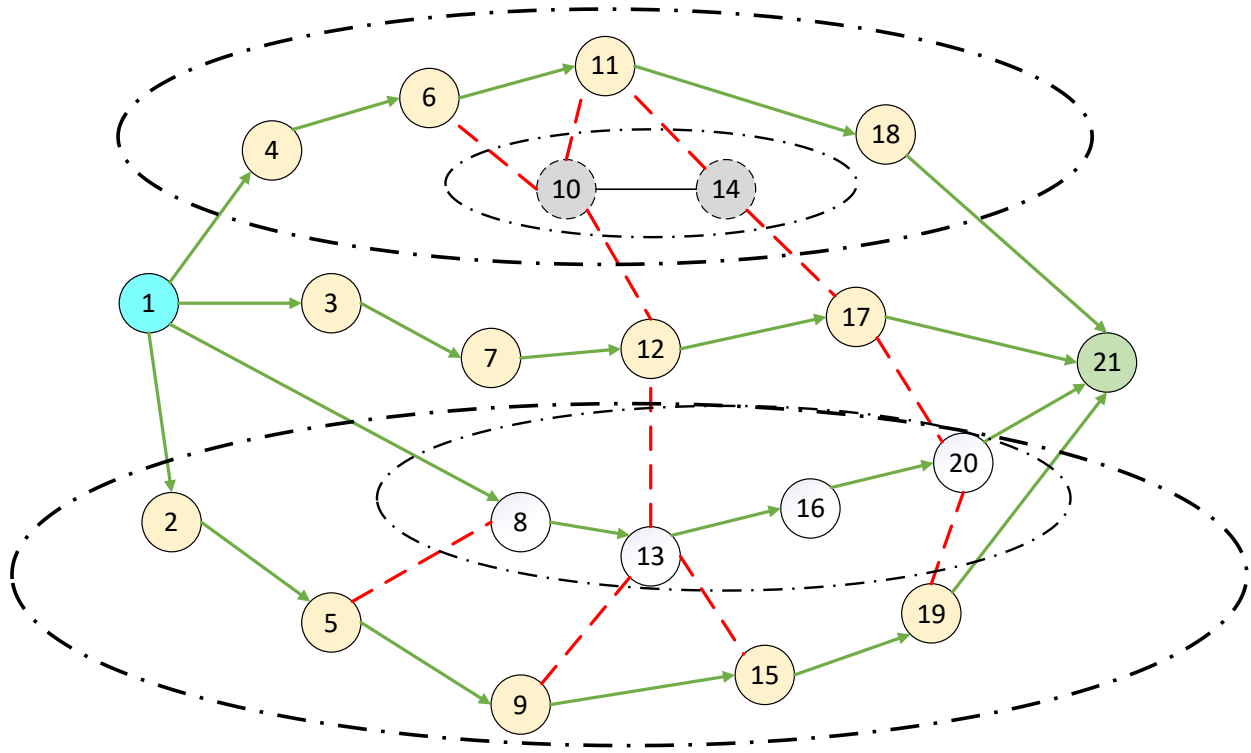


Рис. 2.6. Розбиття топологічного графу мережі на підграфи

Для першого шляху через вершини 3 і 17 зміни часової складності не відбулося, оскільки він будується, базуючись на переборі усього графу. Відповідно, його часова складність  $21^2 = 441$ .

Перший шлях, побудований в середині графа, ділить його на два підграфи – умовно називатимемо їх верхнім та нижнім. Верхній граф складається з шести вершин (4, 6, 10, 11, 14, 18), а нижній – з дев'яти (2, 5, 8, 9, 13, 15, 16, 19, 20). Відповідно, другий шлях, що побудований у верхньому підграфі, підбирається з шести вершин, а не з (21-4), як обраховувалося вище. Таким чином, часова складність другого шляху тепер складає

$$t(L_2) = O(N^2 + N_U^2) = 21^2 + 6^2 = 477, \quad (2.26)$$

де  $N_U$  – кількість вершин верхнього підграфу.

Другий шлях розбиває верхній підграф ще на менші дві частини – підграф з вершин, через які шлях проходить, та ті, що знаходяться між ним та першим шляхом (тобто, вершини 10 і 14). Але обидві ці вершини мають зв'язки лише з уже задіяними вершинами, тому є тупиковими і не використовуватимуться у спробах побудови шляхів, що не перетинаються.

Аналогічним чином, як для другого шляху, обрахуємо часову складність пошуку шляхів включно з третім шляхом, що сформований у нижньому підграфі.

$$t(L_3) = O(N^2 + N_U + N_L^2) = 21^2 + 6^2 + 9^2 = 558, \quad (2.27)$$

де  $N_L$  – кількість вершин нижнього підграфу.

Третім шляхом нижній підграф розділений на два менші підграфи. На відміну від верхнього підграфу, тут склалася ситуація, коли ще один шлях може бути побудований з вершин, що залишилися незадіяними і мають зв'язки між собою та із задіяними вершинами. Часова складність побудови четвертого шляху буде дорівнювати 0, тому що перебору не відбувається і шлях складається з єдиного можливого маршруту, що залишився у підграфі. З цієї причини:

$$t(L_4) = t(L_3) = 558. \quad (2.28)$$

При класичному підході часова складність побудови чотирьох шляхів складала майже удвічі більше – 963. Саме тому для вирішення проблеми  $k$  шляхів, що не перетинаються, на топологічному графі комп'ютерної мережі пропонується метод розділення графу на підграфи з подальшим пошуком шляхів у межах підграфів. Швидка побудова маршрутів, що не перетинаються, дозволить здійснювати миттєву зміну основного шляху надсилання трафіку в мережі *SDN* у разі відмов, що стане на пригоді при необхідності динамічної реконфігурації мережі.

### **2.2.3. Обчислення критерію надійності вузлів для зменшення втрат нееластичного трафіку у мережі SDN**

Для підвищення ефективності побудови віртуальних каналів зв'язку в мережі *SDN* у роботі пропонується централізований підхід до маршрутизації

комп'ютерної мережі: завчасна побудова каналів зв'язку контролером та вибір таких із них, що найбільше відповідають заданій якості обслуговування. Для вирішення проблеми вибору такого каналу зв'язку та зменшення відсотку втрат пакетів в умовах передачі нееластичного трафіку пропонується використання додаткової метрики надійності вузлів.

Розглянемо мережу, що програмно визначена, у вигляді неорієнтованого графа:

$$G = (V, E, W, C), \quad (2.29)$$

де  $V$  – непорожня кінцева множина вершин;  $E$  – множина ребер;  $W$  – вагова функція, яка присвоює кожній вершині певне дійсне число з інтервалу  $(0, 1]$ , який позначає імовірність безвідмовної роботи цього вузла;  $C$  – вагова функція, яка присвоює кожному ребру певне дійсне число.

Щоб досягти мінімальних втрат даних при передачі у мережі, необхідно організувати конструювання трафіку відповідним чином. По-перше, є потреба у ефективній маршрутизації, тому необхідно створити множину шляхів, що не перетинаються (див. пункт 2.3). Це дозволяє оптимально організувати розподіл інформації між маршрутами, враховуючи метрики для кожного шляху та тип трафіку, що ним передається.

Обчислення вагової функції  $C$  для графу мережі було представлено у пункті 2.2 (формула 2.12). Така функція, що дозволяє отримати вагові коефіцієнти для оцінки побудови кожного шляху, демонструє узагальнену метрику з урахуванням параметрів якості обслуговування ( $QoS$ ), щоб шлях для будь-якого типу трафіку був якомога коротшим та таким, що не допускає перевантажень каналів зв'язку. В результаті обчислення метрик якості обслуговування зв'язків, формується умовна вагова матриця, яка використовується контролером  $SDN$  для проектування шляхів між початковим і кінцевим вузлом.

У перевантаженій мережі стани з'єднань постійно зазнають змін. Такі зміни важко передбачити, оскільки є ряд причин, які можуть спричинити несподіване навантаження на окрему лінію зв'язку, яка до цього забезпечувала стабільну

передачу трафіку і дозволяла передавати дані при збереженні високого рівня якості обслуговування. Серед цих причин є наступні:

- збої в роботі пристроїв у проміжних ланках мережі;
- раптові навантаження на ключові канали передачі;
- збільшення кількості користувачів мережі.

У першому випадку при збоях в роботі одного з комутаторів чи маршрутизаторів потрібно пересилати пакети на інші канали. При цьому певні з'єднання можуть значно перевантажуватися.

У другому випадку затримки можуть виникати через часті одночасні підключення до одного з ресурсів великої кількості користувачів. Такі збої очікуються через низку неконтрольованих адміністратором факторів, таких як зростання популярності окремих вузлів, підвищена активність користувачів у певний час доби або різні види злому. Мобільні мережі також можуть зіткнутися з проблемою, коли після підключення додаткових абонентів виникають перевантаження певних каналів передачі, однак ці мережі зазвичай більш підготовлені до такого роду проблем.

У минулих пунктах цього розділу запропоновано та обґрунтовано використання модифікованого алгоритму при формуванні маршрутів. Застосовуючи його, можна знайти всі маршрути, що не перетинаються, між парами вузлів. На основі інформації про стан кожного маршруту контролер *SDN* має вибрати маршрут з найменшим інтегральним критерієм, що враховує стани каналів на момент запиту.

Втім, для підвищення якості передачі трафіку контролер має збирати інформацію не лише про стан зв'язків між вузлами. На практиці комутатори та маршрутизатори мають власні архітектурні недоліки, що не дозволяють їм виконувати пересилання пакетів між своїми портами повністю безвідмовно. Контролер *SDN*, як і у випадку з іншими метриками якості обслуговування, має можливість динамічно збирати та оновлювати статистику вузлів та зберігати її у своїй пам'яті. Для цього він буде аналізувати кількість пакетів, що проходять через комутатор, та порівнювати об'єм трафіку, що отриманий вхідним портом, та той,



що був відправлений через вихідний порт. Це відношення пропонується назвати коефіцієнтом надійності вузла.

$$p_i = \frac{e(t)_{out}}{e(t)_{in}}, \quad (2.30)$$

де  $e(t)_{out}$  – кількість пакетів, що надіслана з вихідного порту за одиницю часу;  $e(t)_{in}$  – кількість пакетів, що була прийнята вхідним портом за той самий час.

Нехай представлений деякий маршрут, для якого необхідно вирішити задачу обчислення ймовірності втрат пакетів через ненадійність вузлів без урахування метрик якості обслуговування каналів зв'язку. Базуючись на відношенні кількості пакетів, що були успішно надіслані з вхідного на вихідний порт кожного комутатора, до повної кількості пакетів, що надсилались з першого вузла, можна провести оцінку надійності кожного вузла  $p_i \in (0, 1]$ .

Для обчислення ймовірності втрати пакету  $P(A_i)$  для маршруту з множини шляхів, що не перетинаються, пропонується застосовувати формулу добутку незалежних подій:

$$P(A_j) = 1 - \prod_{i=1}^n (p_i), \quad i = \overline{1, n} \quad (2.31)$$

де  $P(A_j)$  – ймовірність втрати пакету при його передачі по шляху  $j$ ;  $p_i$  – коефіцієнт надійності вузла.

Надійність вузлів не є метрикою якості обслуговування каналів зв'язку, тому вона не сумується в загальну умовну вагову матрицю відповідно до формули 2.12. Вона пропонується до використання як обмеження – і за певних налаштувань мережі за потреби адміністратора може бути використана у сценарії обрання шляху. Наприклад, може бути задане деяке правило, згідно до якого один з альтернативних маршрутів, який має більшу вартість по метрикам якості обслуговування каналів, але меншу ймовірність втрат пакетів через надійність вузлів на певну величину, буде вважатися основним маршрутом. Інший варіант задання правила: маршрут, що має занадто велику ймовірність втрат пакетів, не може бути включений до переліку альтернативних маршрутів і буде ігноруватись контролером для вибору направлення пакетів.

Використання метрики надійності вузла, за рахунок постійного відслідковування контролером *SDN* стану портів комутаторів, дозволяє також оперативно реагувати на відмови власне вузлів, а не лише каналів зв'язку. У разі виходу з ладу кількох портів певного комутатора, контролер має змогу одразу тимчасово виключити з топологічного малюнку мережі пов'язані з цими портами зв'язки, а у разі виходу з ладу усіх портів одразу – повністю виключити вузол з графу мережі з усіма зв'язками, що йдуть до та з нього. Це дозволить сприймати вузол як аварійну ділянку та уникати його при передачі трафіку, доки роботу комутатора не буде знову відновлено.

## Висновки до розділу 2

Існуючі алгоритми не забезпечують комплексне вирішення задачі побудови віртуальної мережі з визначенням шляхів, які би забезпечували з'єднання від вихідної точки до адресата, враховуючи надійність комутаторів та стан зв'язків між ними. Для забезпечення якомога менших часових витрат на передачу трафіку при побудові маршруту між вузлами запропоновано одночасно враховувати чотири метрики якості обслуговування: кількість переходів між вершинами, доступну смугу пропускання, часову затримку та відсоток втрачених пакетів. При створенні запиту на визначення маршруту для трафіку у мережі статус вузлів оновлюється з метою побудови шляху, що відповідає якості обслуговування.

У розділі запропонований інтегральний критерій для побудови каналів зв'язку з урахуванням якості обслуговування у віртуальній комп'ютерній мережі на основі технології *SDN*. Він представляє собою комбінацію умовних вагових матриць (кожна побудована для «свого» параметру обслуговування) для формування єдиної матриці переходів між вузлами. Залежно від типу трафіку, для кожного параметру якості обслуговування мережі задаються спеціальні коефіцієнти пріоритезації. Використання інтегрального критерію дозволяє досягти збереження швидкості передачі даних для будь-якого типу трафіку та збільшити гнучкість побудови каналів зв'язку у мережі при змінах її конфігурації.

У розділі продемонстровано, що комбінаторний алгоритм багатошляхової маршрутизації, що формує  $k$  шляхів, що не перетинаються, на графі мережі, характеризується меншою часовою складністю в порівнянні з базовим алгоритмом у випадку розділення графа на підграфи з виконанням комбінаторних операцій в межах кожного з них окремо. Розділення графа мережі на підграфи зменшило часову складність побудови каналів зв'язку з використанням інтегрального критерію.

У розділі запропонований критерій надійності вузлів мережі для побудови каналів зв'язку. Критерій надійності є числовим відношенням між кількістю пакетів, отриманих комутатором, до кількості, що була надіслана ним з вихідного

порту за ту саму одиницю часу. За допомогою порівняльного аналізу вхідного та вихідного трафіку на портах комутаторів мережі *SDN* визначається ймовірність втрати пакетів при проходженні вузлів на кожному з альтернативних маршрутів. Використання критерію надійності вузлів дозволило збільшити надійність передачі трафіку по каналам зв'язку та зменшити втрати даних.

У даному розділі запропонований метод побудови каналів зв'язку віртуальної мережі на основі технології *SDN*, який використовує інтегральний критерій побудови каналів зв'язку мережі з урахуванням якості обслуговування, критерій надійності вузлів мережі та розділення її на підмережі під час процесу побудови каналів зв'язку. Метод дозволив зберегти швидкість, зменшити затримку та втрати даних при їхній передачі для будь-якого типу трафіку у разі виходу з ладу окремих ділянок мережі.

## РОЗДІЛ 3

## УДОСКОНАЛЕННЯ МЕТОДУ ДИНАМІЧНОЇ РЕКОНФІГУРАЦІЇ ВІРТУАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN

### 3.1. Реалізація методу побудови каналів зв'язку на основі інтегрального критерію з урахуванням якості обслуговування

Під час функціонування мережі *SDN* можливе раптове відключення частини комутаторів. Це обумовлює необхідність реконфігурації – перебудови доступних для передачі даних каналів зв'язку. Відомі реалізації контролерів *SDN* зазвичай перебудовують шляхи передачі даних з нуля, що веде до великих часових витрат на реконфігурацію. Використовуючи запропонований у другому розділі метод побудови каналів зв'язку на основі інтегрального критерію та пам'ять контролера, яка зберігає множину раніше побудованих шляхів (в тому числі, недоступних для передачі після появи аварійних ділянок мережі), процес динамічної реконфігурації можна удосконалити.

Реконфігурація мережі починається з побудови каналів зв'язку згідно графа топології мережі, які використовуватимуться в процесі передачі трафіку між початковою та кінцевою вершинами (рис. 3.1).

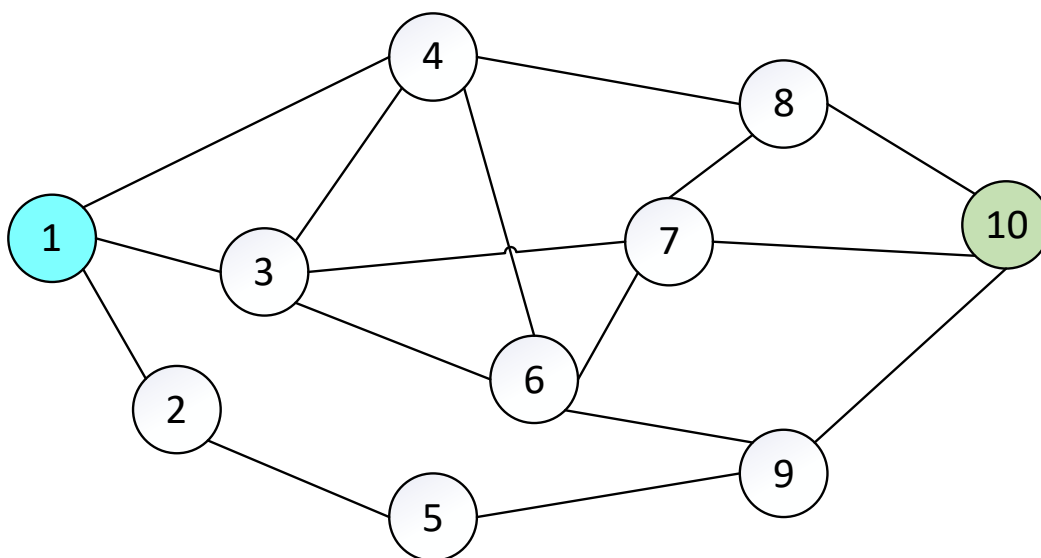


Рис. 3.1. Граф топологічної організації мережі

Як зазначалося у розділі 2, обов'язковою вимогою до архітектури мережі є врахування параметрів якості обслуговування маршрутів. В експериментальному графі 10 вузлів. Передбачається, що вузол 1 є вихідною точкою, а до точки 10 необхідно побудувати шлях.

В ході своїх обчислень контролеру необхідно сформувати множину маршрутів, що не перетинаються, від вихідної вершини до адресата, або дійти висновку, що встановити зв'язок між вузлами неможливо (рис. 3.2) [73].

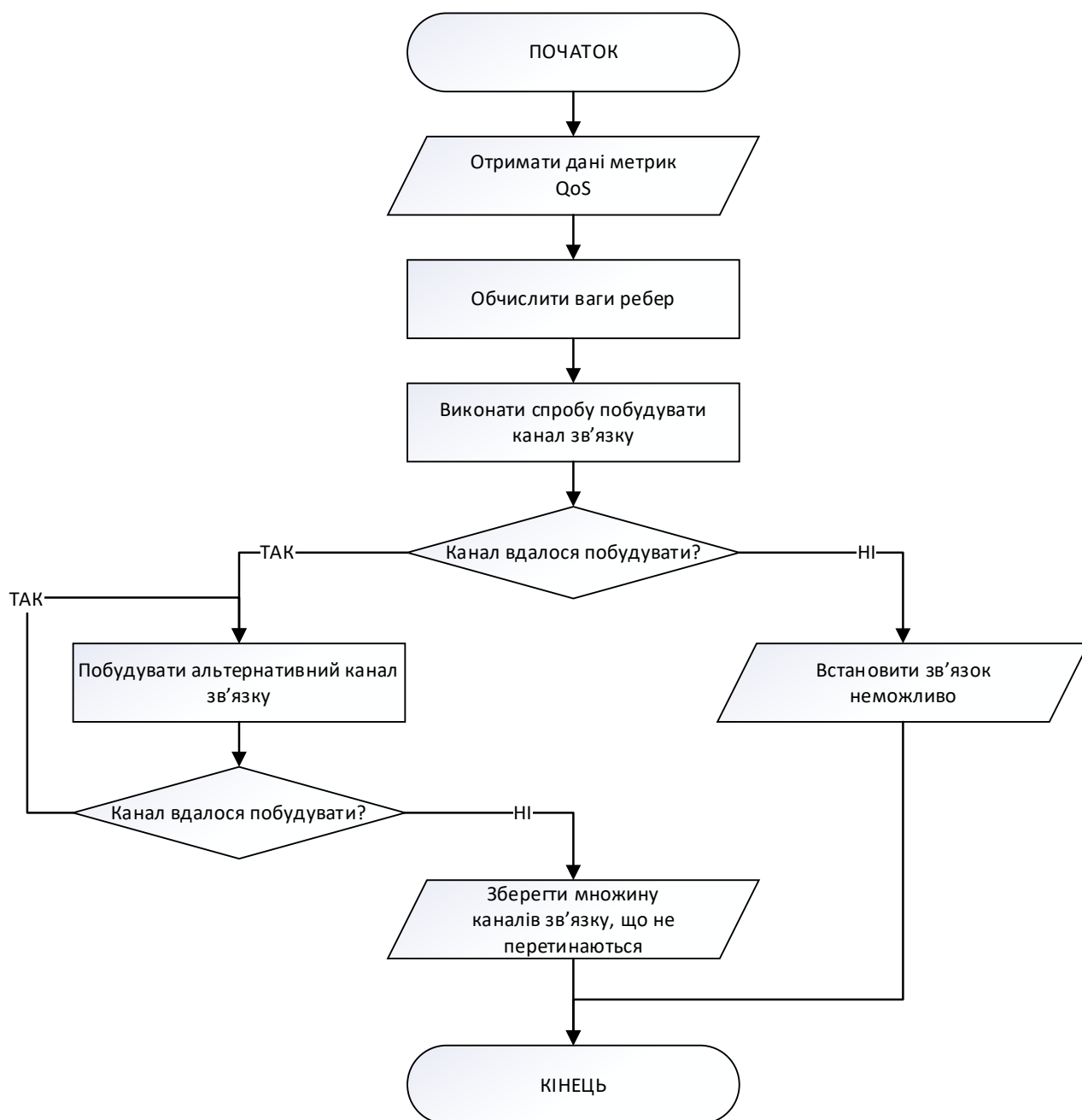


Рис. 3.2. Алгоритм побудови множини каналів зв'язку, що не перетинаються

Згідно алгоритму Дейкстри, прапорець нинішньої точки на початку встановлюється на вершину 1, відстань від початкової точки до власного вузла встановлюється на 0, а інші вузли позначаються тегами. Усі десять вузлів, що позначені на графі, поміщаються до множини вузлів шляху  $S$ .

Першою з метрик, що враховуються, є кількість переходів між вершинами (хопів). Для оцінки відстані між вершинами згідно неї стандартним підходом є надання одиничної ваги кожному ребру графа (рис. 3.3).

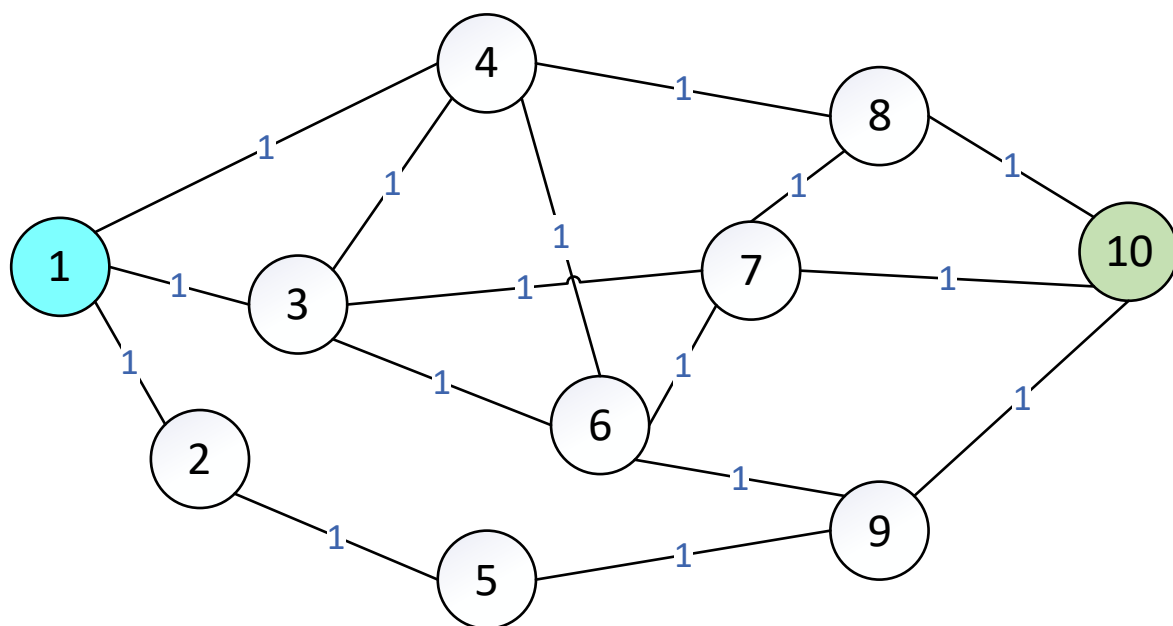


Рис. 3.3. Граф мережі із ребрами, оціненими за метрикою кількості переходів

На практиці у мережах загального користування за «перехід» між пристроями можуть вважатися інші показники. Наприклад, частина пристроїв можуть вважатись пріоритетними за рахунок більш потужного апаратного забезпечення. В такому випадку адміністратор може задати метрику таким чином, щоб скерувати трафік на такі пристрої задля забезпечення стабільнішої передачі трафіку.

Наступна метрика – доступна смуга пропускання. Згідно формули 2.2, вона є обернено пропорційною завантаженості маршруту. При вирішенні задачі побудови віртуальної мережі метрика пропускну здатності для каналу зв'язку обчислюється наступним чином: контролер *SDN*, засновуючись на даних про пропускну здатність каналів зв'язку у пакетах на секунду, введених

адміністратором, обчислює метрику як округлене число від ділення різниці величини цієї пропускної здатності каналу з максимально можливою технічною пропускною можливістю каналу (нехай вона становить 1000 пакетів на секунду) на 100.

В рамках вирішення поставленої у даному пункті задачі вважатимемо прямий маршрут між початковою і кінцевою вершинами найменш широким (тому оціненим як більш вартісний). З найбільшою смугою пропускання трафіку доступні проміжні маршрути всередині графу (рис. 3.4).

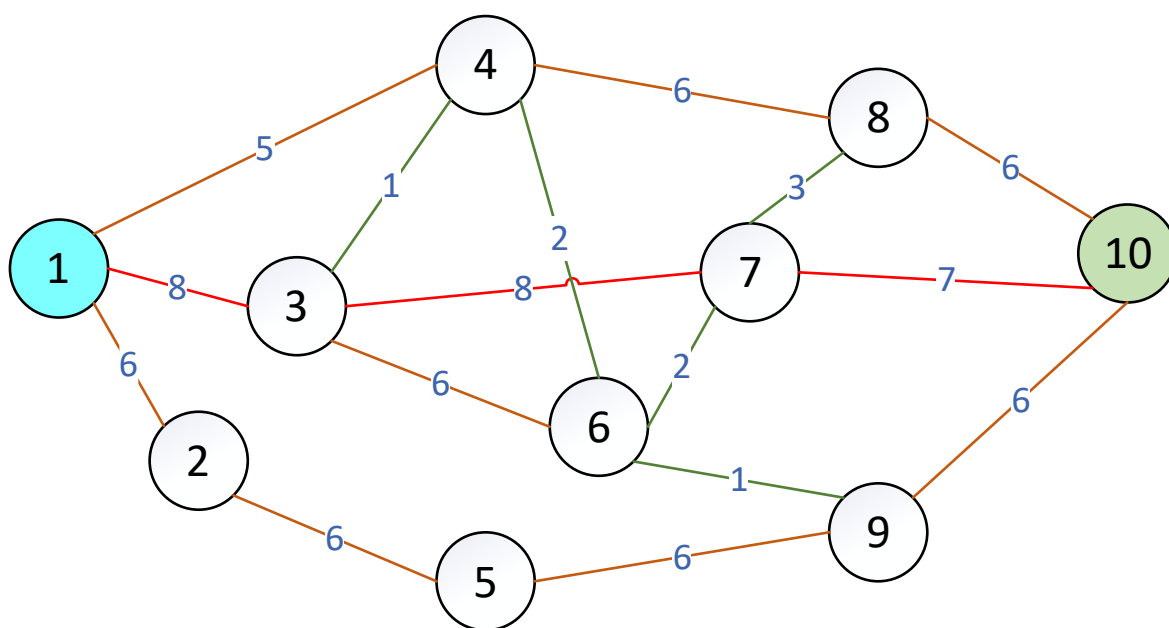


Рис. 3.4. Граф мережі із ребрами, оціненими за метрикою доступної смуги пропускання

Коли організація каналів зв'язку заснована лише на відстанях між проміжними пристроями, то використання найбільш коротких маршрутів спочатку буде виконувати передачу з прийнятливими показниками, але при збільшенні об'ємів трафіку вузькі переходи швидко стануть перевантаженими, що погіршить стабільність передачі даних у мережі.

Далі пропонується до розгляду метрика часової затримки переходів. Вона може бути зібрана лише на основі даних, що були отримані контролером з попереднього функціонування мережі хоча б певний проміжок часу. Тому при



запуску *SDN* маршрутизації на новій інфраструктурі та топології умовна матриця затримки буде тимчасово не обрахована та рівна 0.

Для випадку, що розглядається, уявімо, що за вимогою власника мережі для оцінки вартості зв'язків використовуються цілі числа від округлення десятої долі завантаженості маршрута у відсотках. Відносно прямий маршрут між вершинами 1 і 10 сильно завантажений, обхідні через вершини 4 і 8, а також 2 і 9 – середньо завантажені. Простоюють проміжні зв'язки всередині графа. Така ситуація доволі часто наявна у реальних комп'ютерних мережах загального користування, де існують основні магістралі, по яким направляється більша частина трафіку, у той час як здатні до проходження великої кількості трафіку проміжні зв'язки простоюють (рис. 3.5).

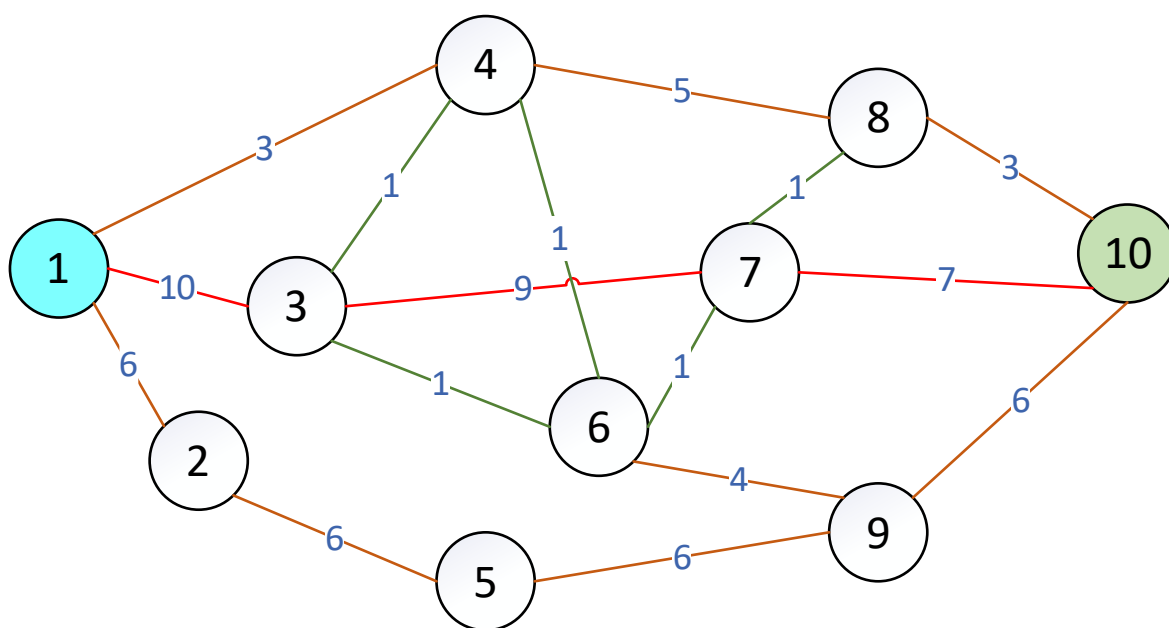


Рис. 3.5. Граф мережі із ребрами, оціненими за метрикою часової затримки

Останньою метрикою, що пропонується враховувати при організації каналів зв'язку, є відсоток втрачених пакетів. Як і у випадку для завантаженості маршрутів, вона може бути обчислена лише після функціонування мережі протягом певного часу, тому при безпосередньому запуску маршрутизації на новій топології умовна матриця втрачених пакетів дорівнюватиме нулю. Нехай для проектування існує вимога цілочислового представлення відсотку втрачених

пакетів на даному каналі зв'язку. Втрати безпосередньо пов'язані із завантаженням та пропускною здатністю каналу. Нехай на завантаженому центральному шляху втрати пакетів досягають 8%, але інші зв'язки демонструють в цілому стабільну передачу трафіку (рис. 3.6).

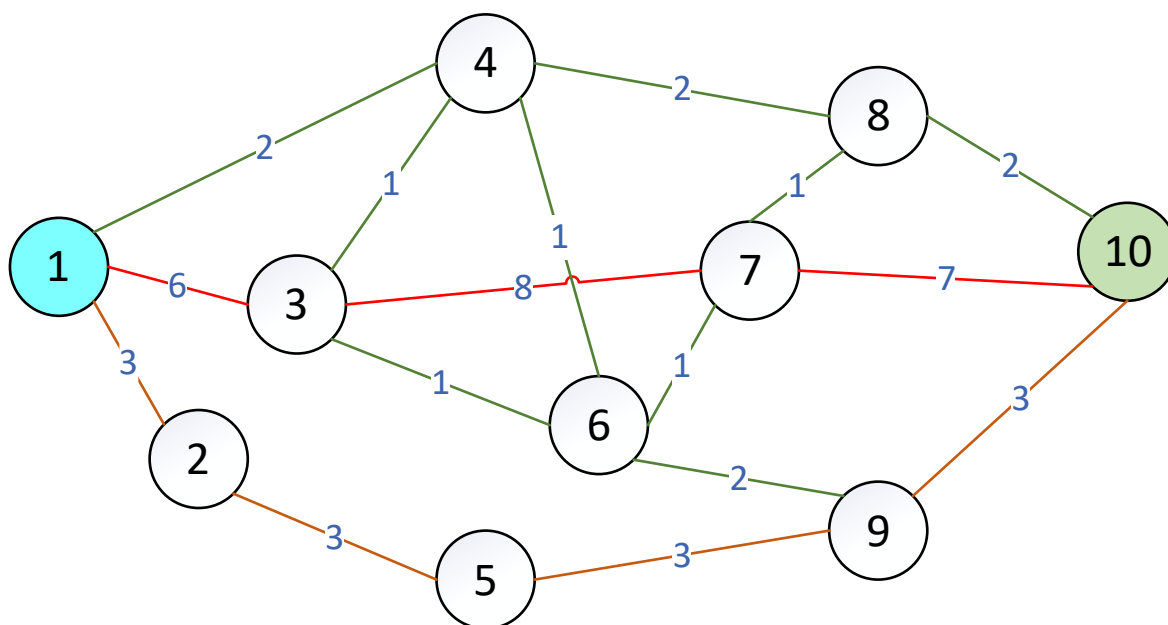


Рис. 3.6. Граф мережі із ребрами, оціненими за метрикою відсотка втрачених пакетів

На основі описаних матриць вагів переходів, можна спрогнозувати, які шляхи передачі трафіку будуть обрані алгоритмом Дейкстри для маршрутизації набору даних, що необхідно надіслати до вершини 10 при даному стані мережі. Перевантажений центральний маршрут  $1 \rightarrow 3 \rightarrow 7 \rightarrow 10$  був би обраний лише при врахуванні однієї метрики кількості переходів, але, очевидно, він не є підходящим згідно якості обслуговування, оскільки через велику завантаженість на ньому існує велика ймовірність втрат даних.

Нехай власником мережі задана висока пріоритетність уникання каналів з великою часовою затримкою – коефіцієнт пріоритезації цієї метрики встановлено у значення 10. Доволі важливими також вважаються метрики кількості переходів та відсотка втрачених пакетів (коефіцієнти дорівнюють 5). Смуга пропускання згідно вимогам має вважатися найменш пріоритетною (коефіцієнт дорівнює 2).

На основі отриманих даних про вимоги щодо коефіцієнтів пріоритезації та метрик стану мережі на момент генерації набору даних, який необхідно надіслати, контролер *SDN* має виконати обчислення узагальненої вагової матриці на основі формули 2.12 наступним чином:

$$K_{ij} = \sum (5H_{ij} + 2B_{ij} + 10D_{ij} + 5L_{ij}), \quad (3.1)$$

де  $H_{ij}$ ,  $B_{ij}$ ,  $D_{ij}$  і  $L_{ij}$  – вагові матриці ребер графу, що представлені на рис. 3.3–3.6.

Відповідно, топологічний граф мережі, побудований на основі об'єднаної матриці вартості каналів зв'язку  $K_{ij}$  після обчислень контролера виглядатиме як показано на рис. 3.7.

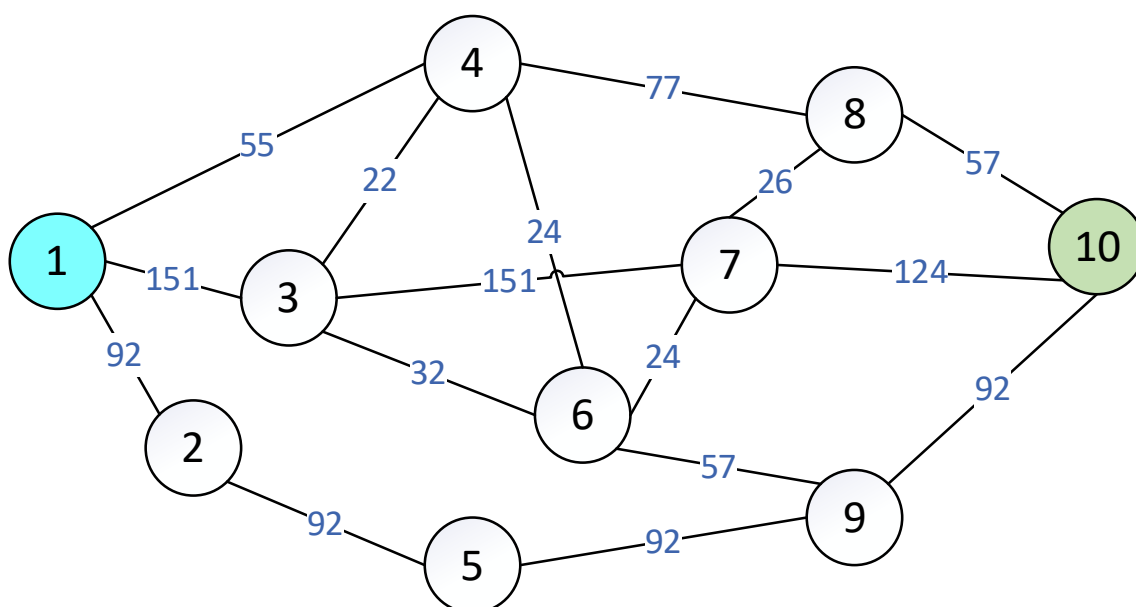


Рис. 3.7. Граф мережі із ребрами, оціненими за чотирма метриками якості обслуговування

Такий вигляд графа є графічним уявленням топології мережі та зв'язків у ній, на базі чого працює алгоритм Дейкстри. Просумовані ваги ребер являють собою математичне представлення пріоритезації метрик.

В ході роботи алгоритму та вибору найменшої ваги наступного переходу формується маршрут, що використовує «проміжні» переходи, що були не задіяні в ході передачі даних без урахування якості обслуговування. Таким чином, завдяки обчисленням був сформований новий віртуальний канал зв'язку  $1 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 10$  (рис. 3.8).

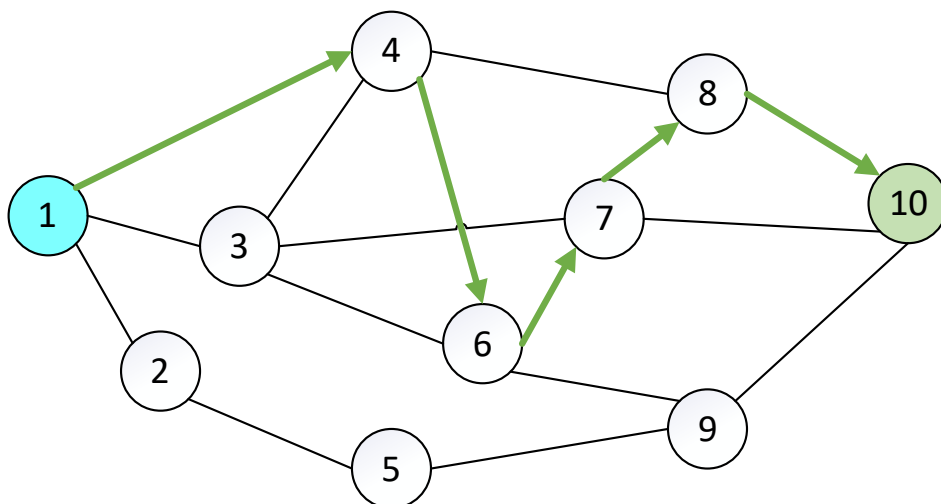


Рис. 3.8. Перший сформований канал зв'язку між вершинами 1 і 10 на основі інтегрального критерію з урахуванням якості обслуговування

Після побудови першого (як і після будь-якого чергового) маршруту з графу топології мережі для подальших обчислень вилучаються вершини, що були пройдені у минулих маршрутах та зв'язки до них. Граф таким чином ділиться на підграфи (вже відомі маршрути виступають як переріз), в межах яких і відбувається подальший пошук маршрутів для поповнення множини шляхів.

У випадку задачі пошуку шляхів для побудови мережі, що вирішується у цьому пункті, найбільш підходящий згідно метрик якості обслуговування маршрут проходить через вершини 4, 6, 7 і 8, тому для побудови інших маршрутів, що не перетинаються із ним, вони виключені з подальшої побудови (рис. 3.9).

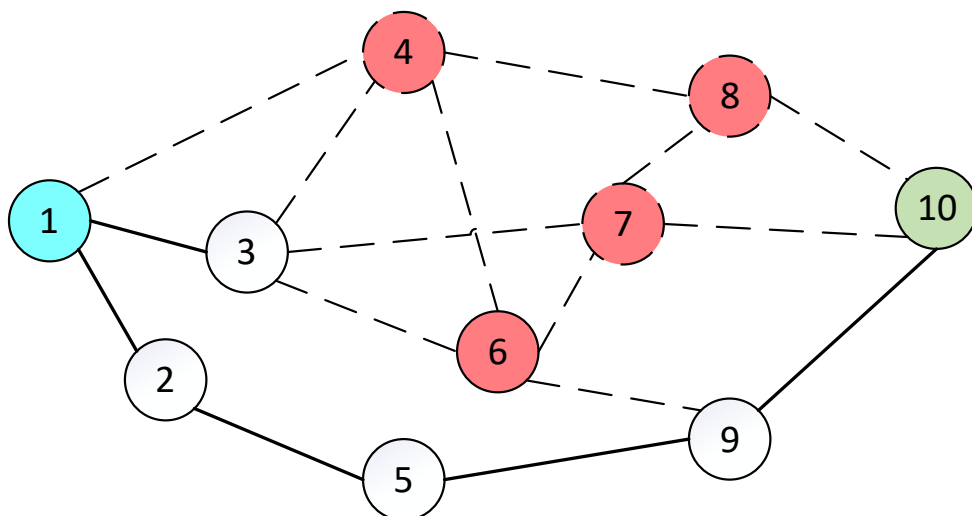


Рис. 3.9. Побудова залишкового підграфу мережі після побудови першого каналу зв'язку між вершинами 1 і 10

Через вершини, що залишилися, існує лише один можливий шлях з вузла 1 до 10, і він обирається як єдина альтернатива. Для такого випадку можна стверджувати, що урахування параметрів якості обслуговування для цього маршруту було виконано неявно, оскільки цей маршрут був побудований як безальтернативний після побудови інших каналів зв'язку. Аби подолати невизначеність відповідності таких шляхів поставленій задачі ефективної передачі трафіку та сформувати єдиний рівень пріоритетності усіх маршрутів, що не перетинаються, знадобиться додаткова пріоритезація. У пункті 2.4 запропонований критерій вибору шляху по метриці надійності вузлів через обчислення ймовірності втрати пакету при його транспортуванні з вхідного на вихідний порт кожного пристрою. Саме за рахунок цього обмеження можна привести у відповідність маршрути, що побудовані за «надлишковим принципом», тобто залишилися як безальтернативні після поділення графа топології мережі на підграфи.

Після виключення проміжних вершин, через які проходить другий маршрут, на графі залишається лише одна тупикова вершина 3, через яку побудувати маршрут до вузла 10 неможливо. Таким чином, в ході роботи комбінаторного алгоритму формуються два шляхи, що не перетинаються, між вершинами 1 і 10 (рис. 3.10).

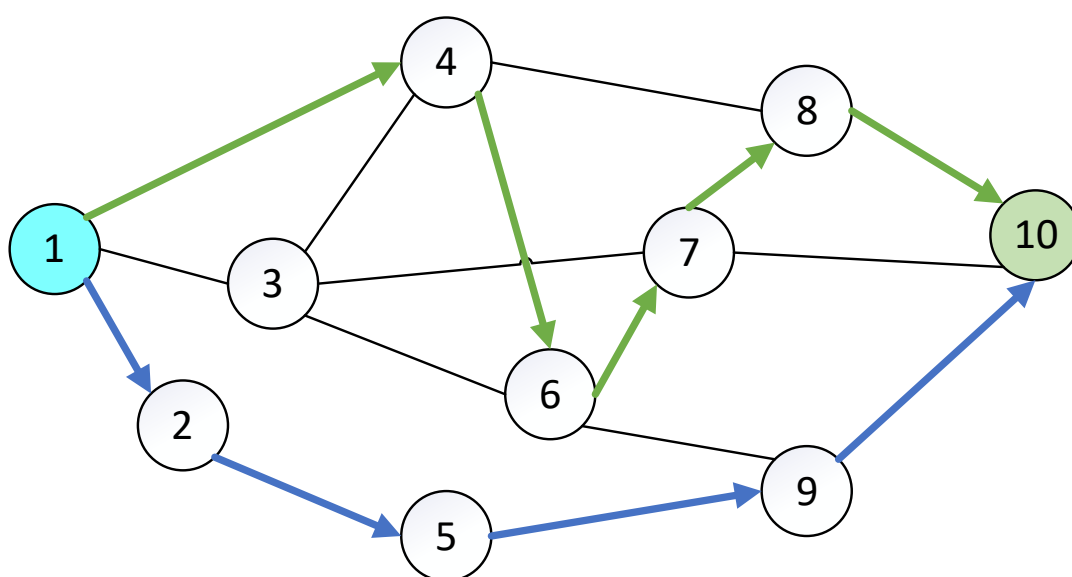


Рис. 3.10. Канали зв'язку між вершинами 1 і 10, побудовані на основі інтегрального критерію з урахуванням якості обслуговування

Отже, використовуючи запропонований метод побудови каналів зв'язку на базі інтегрального критерію, були отримані маршрути, що здатні вирішити проблему ефективної передачі трафіку та враховувати статистичні дані, що динамічно збираються контролером *SDN* у реальному часі. Для реагування на зміну вимог до мережі адміністратори можуть змінити необхідний рівень пріоритетності для кожної з метрик *QoS*. Програмне забезпечення контролера розглядатиме вибрані рівні як коефіцієнти пріоритезації для розрахунку чотирьох нових умовних вагових матриць для топології мережі. Опісля контролер виконає їхнє сумування відповідно до встановлених рівнів, щоб сформувати повну матрицю вагів ребер графу, а потім з використанням алгоритму Дейкстри та принципу розділення графу топології мережі на підграфи побудує множину віртуальних каналів зв'язку, що не перетинаються, між вершиною-джерелом та вершиною-адресатом.

### **3.2. Пріоритезація каналів передачі даних на основі критерію надійності вузлів мережі**

Наступним процесом, що відбувається в рамках динамічної реконфігурації після побудови каналів зв'язку, є пріоритезація побудованих каналів, тобто вибір тих, які проходять через вузли, що відповідають заданому критерію надійності.

Метрика надійності вузла  $p_i$  базується на розрахунку ймовірності втрати пакету при його переносі від вхідного до вихідного порту кожного програмно-визначеного комутатора. Вона представляється у числовому вираженні  $p_i \in (0, 1]$  як відношення пакетів, що успішно транспортовані через пристрій, до загальної кількості тих, що надійшли до нього.

Щоб ефективно вирішити проблему передачі трафіку та узгодити пріоритети всіх маршрутів, потрібен процес пріоритезації (пункт 2.4), алгоритм якого зображений на рис. 3.11.



Рис. 3.11. Алгоритму відбору каналів зв'язку по критерію надійності вузлів

Задати величину надійності вузлів можна на основі раніше зібраних контролером *SDN* даних про стан функціонування мережі. Згідно формули 2.29 формуються коефіцієнти надійності для кожного вузла та оновлюються через заданий проміжок часу, якщо через вузол продовжується передача даних та є можливість оновлювати статистичні дані. Інший спосіб: задання величини

надійності на основі вимог, обраних власником або адміністратором мережі, що, у свою чергу, можуть базуватись на апаратних та програмних недоліках конкретного комутатора. В такому разі встановлені коефіцієнти надійності для вузлів мережі використовуються як обмеження для доступності маршрутів.

Обидва підходи до встановлення значень надійності вузлів дозволяють співставляти шляхи, які були створені як безальтернативні – тобто в умовах зменшення масштабу підграфів були єдиними можливими маршрутами між вихідним вузлом та вузлом-адресатом. Це дозволить виконати вибір шляху в умовах невизначеності, коли врахування параметрів якості обслуговування каналів зв'язку присутнє лише неявним чином (через побудову інших шляхів).

Запропонований метод побудови шляхів за рахунок врахування метрики надійності вузлів можна використати для вирішення задачі вибору шляху для графу топології мережі. Контролер *SDN* володіє статистикою проходження пакетів через пристрої, зібраної на основі минулого функціонування мережі. Таким чином, для кожного комутатора контролером виставлений коефіцієнт його надійності у числовому вираженні. Граф топології мережі з зображенням двох шляхів, що не перетинаються, які були побудовані у минулому пункті, та проставлених коефіцієнтів надійності виглядатиме так, як показано на рис. 3.12.

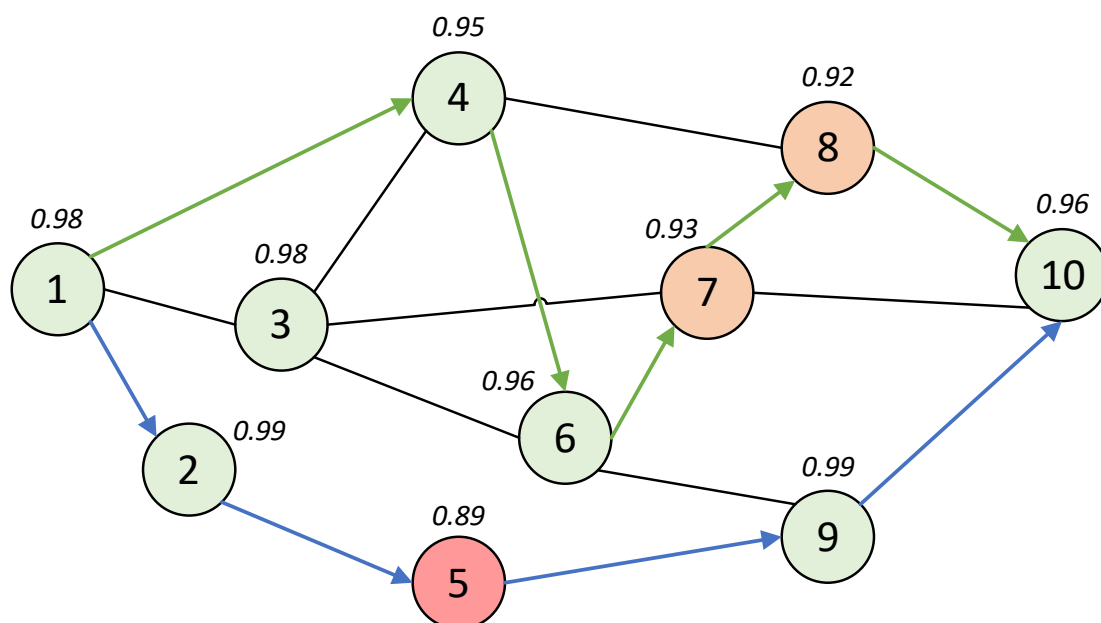


Рис. 3.12. Граф мережі з каналами зв'язку між вершинами 1 і 10 та коефіцієнтами надійності вузлів



На основі зібраної контролером інформації, можна перейти до обчислень ймовірностей втрат пакетів через недоліки апаратного чи програмного забезпечення комутаторів вже безпосередньо для побудованих раніше маршрутів. Коефіцієнти надійності є числовим вираженням ймовірності втрати пакету на конкретному вузлі, тож їх можна вважати незалежними подіями. Відповідно, для обчислення загальної ймовірності втрати пакету на усьому маршруті згідно формули 2.30 необхідно отримати добуток коефіцієнтів надійності для усіх вузлів (включаючи початковий та кінцевий), через які проходить маршрут.

Застосувавши цей метод для задачі, що вирішується у даному розділі, отримуємо ймовірність втрати пакету через ненадійність комутаторів для першого маршруту ( $1 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 10$ ), що був визначений як найбільш підходящий при врахуванні лише параметрів якості обслуговування для зв'язків, майже на 50% більшу, ніж для другого ( $1 \rightarrow 2 \rightarrow 5 \rightarrow 9 \rightarrow 10$ ).

$$P(A_1) = 1 - (0,98 \cdot 0,95 \cdot 0,96 \cdot 0,93 \cdot 0,92 \cdot 0,96) \approx 0,266 \quad (3.2)$$

$$P(A_2) = 1 - (0,98 \cdot 0,99 \cdot 0,89 \cdot 0,99 \cdot 0,96) \approx 0,18 \quad (3.3)$$

Це означає, що при врахуванні метрики надійності вузлів другий маршрут має бути обраний як пріоритетний через меншу ймовірність втрати даних при їх передачі по ньому. Причина, за якої маршрут виявився менш надійним, криється у більшій довжині маршруту. Для мереж великого масштабу, де наявні маршрути через багато перевантажених вузлових точок, це є вкрай актуальною проблемою, тому для них метод, що пропонується, дозволить досягти збільшення надійності передачі даних.

У даному методі пропонується використовувати статистику надійності вузлів не лише у якості пріоритезації, а й у якості обмеження. Так, обмеження можуть бути задані двома способами в залежності від типу трафіку, що планується надсилати. Перший спосіб: по надійності шляху – маршрут із множини шляхів, що не перетинаються, на якому ймовірність втрати пакету при його повному проходженні не відповідає мінімальному значенню за вимогами, виключається з множини альтернативних маршрутів. Другий спосіб: по надійності окремих вузлів – вузли, які мають коефіцієнт надійності менший, ніж мінімальний за вимогами,

виключаються з топологічного графу мережі, тобто усі шляхи, що проходять через такі вузли, мають бути виключені з множини альтернатив.

При задіянні обмеження по надійності шляху  $P(A_j)=0,2$  для задачі пошуку шляхів на рис. 3.12 і певного типу трафіку, перший маршрут не може бути використаний для передачі, оскільки  $P(A_1)>P(A_j)$ . Таким чином, другий маршрут залишається безальтернативним і використовуватиметься для передачі трафіку цього типу.

При задіянні обмеження по надійності окремих вузлів  $p_i=0,9$  для тієї ж задачі пошуку, безальтернативним залишиться перший маршрут: не дивлячись на те, що при об'єднанні ймовірностей втрат пакетів як незалежних подій цей маршрут в цілому є менш надійним, альтернативний йому другий маршрут проходить через вершину 5, що має незадовільну з точки зору вимог надійність, тому що  $p_5 < p_i$ . Ця ненадійна вершина буде вилучена зі списку доступних для передачі, у зв'язку з чим другий маршрут перестає бути підходящим для передачі даних (рис. 3.13).

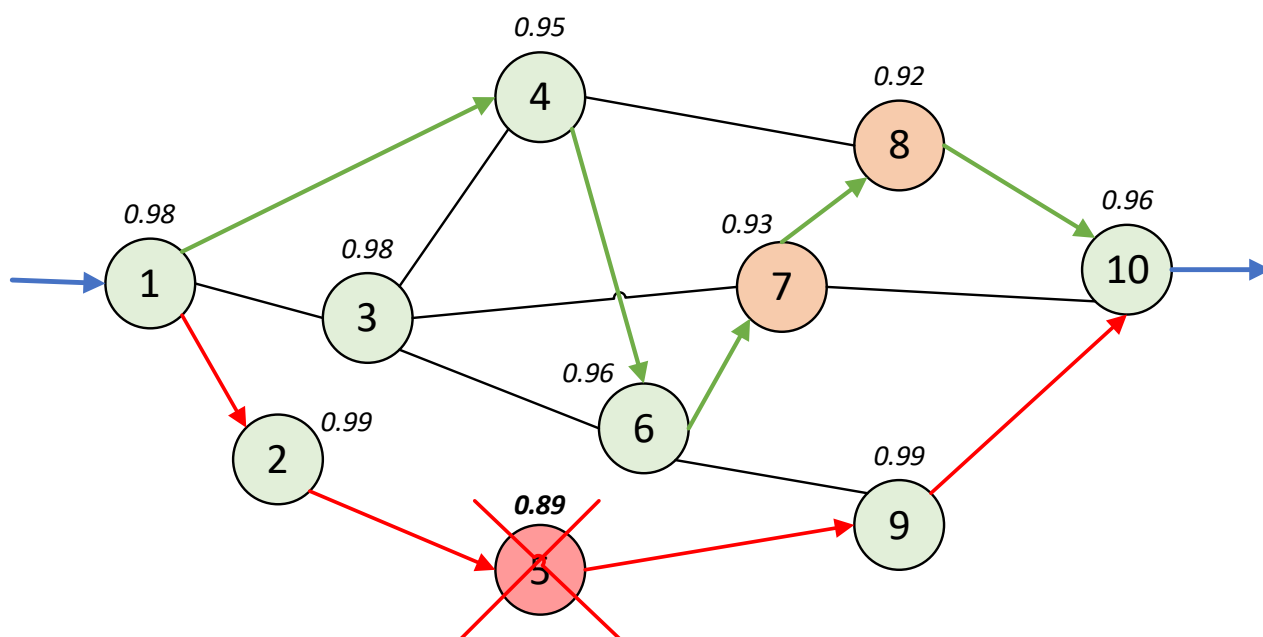


Рис. 3.13. Виключення вершини, що не відповідає критеріям надійності, з процесу пріоритезації каналів зв'язку

Нехай для даної задачі адміністратором обрано обмеження по надійності окремих вузлів. В такому випадку маршрут  $1 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 10$  залишається єдиною альтернативою для передачі даних.

### **3.3. Метод динамічної реконфігурації віртуальної мережі на основі технології SDN з використанням резервних каналів зв'язку**

Необхідність реконфігурації мережі виникає через ряд чинників. По-перше, у зв'язку із можливим швидким збільшенням обсягу даних та кількості користувачів, що їх передають, деякі вузли можуть почати перевантажуватись та виходити з ладу. Це особливо часто відбувається у мобільних мережах, здебільшого у тих, що мають велике географічне покриття.

По-друге, мережа може періодично розширюватись, через що вже існуючі вузли приєднуються до нових. Раніше побудовані шляхи передачі даних можуть стати неактуальними через зміну топологічного малюнку мережі. Щоб уникнути негативного впливу на швидкість передачі даних або навіть тимчасової зупинки цього процесу, необхідно своєчасно провести процес конвергенції комп'ютерної мережі, тобто перебудову каналів зв'язку.

Найкращим способом пришвидшити реконфігурацію є автоматизація цього процесу. Протоколи та контролери, створені на основі технології програмно-визначених мереж, дозволяють проводити ремаршрутизацію за відрізки часу, необхідні для базових процесорних обчислень, що на порядки швидше, ніж ручне переналаштування мережі. Крім того, контролер *SDN*, який архітектурно є комп'ютером з можливістю встановлення широкого набору різноманітних додатків для управління мережею, надає можливість зберігати топологію мережі в усьому її обсязі. Це дозволяє йому виконати переналаштування окремих вузлів або каналів зв'язку з урахуванням аналізу впливу нових налаштувань на мережу в цілому.

Існуючі методи реконфігурації *SDN* контролерів не використовують можливість збереження маршрутів передачі у пам'яті. В даному розділі запропоновано використовувати резервні канали зв'язку у випадку недоступності основного, що є останнім етапом реконфігурації мережі.

Етапи запропонованого методу динамічної реконфігурації віртуальної мережі *SDN*, згідно яких здійснюється вибір резервного шляху, зображені на рис. 3.14.

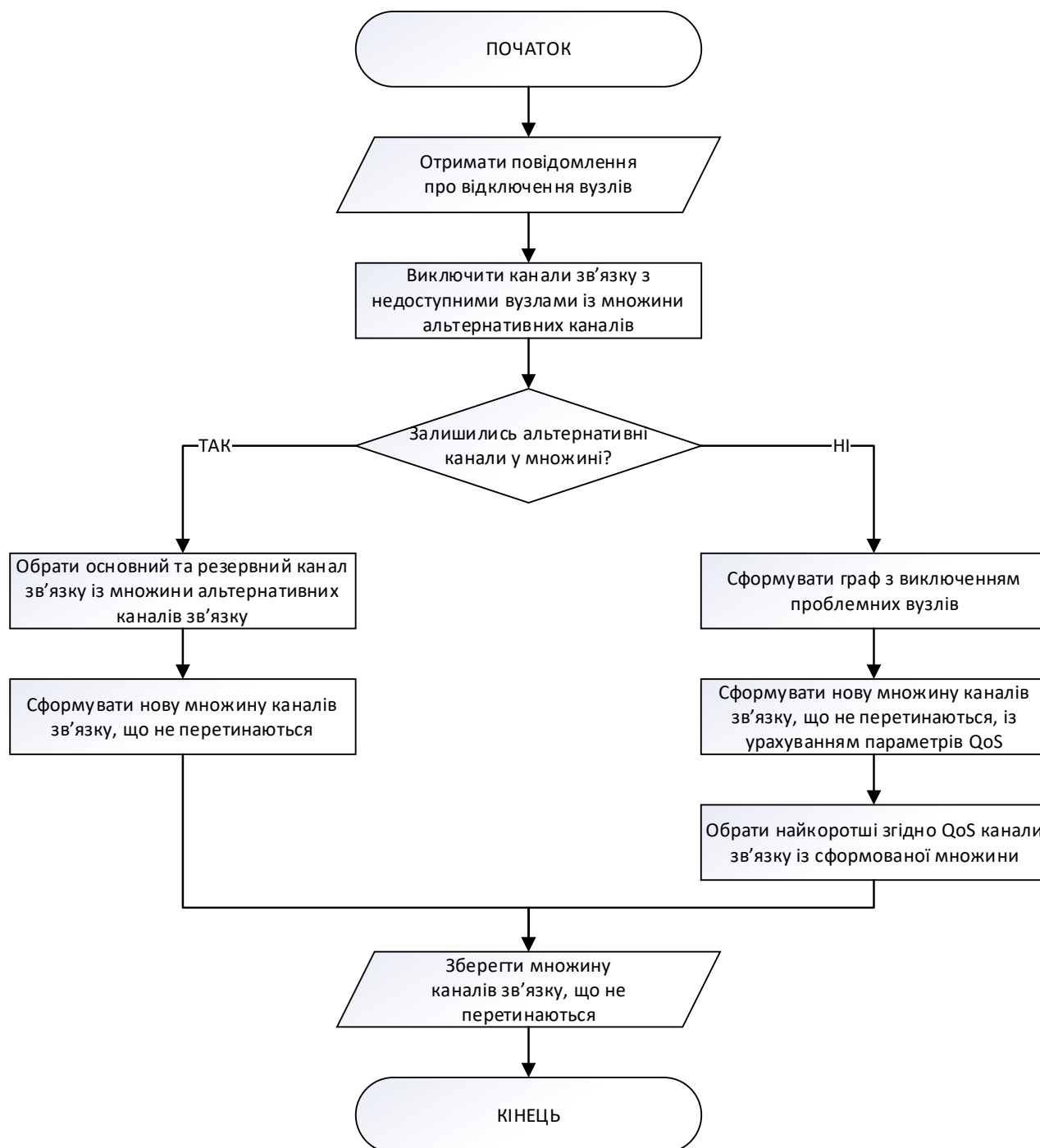


Рис. 3.14. Етапи методу динамічної реконфігурації мережі *SDN*

Нехай у мережі, зображеній на рис. 3.13, були знижені вимоги до надійності вузлів до  $p_i = 0,88$ . В цьому випадку, замість перебудови нових каналів зв'язку, контролер встановить маршрут  $1 \rightarrow 2 \rightarrow 5 \rightarrow 9 \rightarrow 10$  як резервний (оскільки його сумарна вага більша, ніж у основного маршруту  $1 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 10$ ). Часова складність побудови каналів зв'язку з точки зору топології мережі дорівнює нулю. Часові витрати на збереження резервного шляху визначаються лише часом, який

необхідний контролеру для запису в пам'ять оновленої множини каналів зв'язку. Доступні шляхи для передачі даних відповідатимуть рис. 3.12.

Для виникнення причини для проведення нової реконфігурації припустимо, що комутатори 4, 5 і 6 були вилучені з фізичної топології та більше не можуть виконувати транспортування пакетів. При цьому, мережа під час їхнього вилучення продовжувала функціонувати, а за вимогами, вона має виконувати передачу даних між вершинами 1 і 10 без перебоїв для зручності користувачів.

Як тільки вузли мережі перестали передавати трафік, контролер *SDN* отримує повідомлення про це, після чого розпочинає процес реконфігурації мережі для перенаправлення трафіку. Перше, що він перевіряє – можливість вибору альтернативного шляху з множини шляхів, що були побудовані раніше. Перший із списку шляхів, що відповідають якості обслуговування, буде обраний для передачі трафіку. Якщо ж альтернативних шляхів немає, то відбувається процес пошуку нових обхідних маршрутів. З технічної точки зору, він є тим самим процесом побудови каналів зв'язку, що запропонований у минулому розділі, але для графу, що вже включає в себе інформацію про раніше визначені особливості мережі. Вершини 4, 5 і 6, а також усі зв'язки до них на графі відсутні через недоступність цих вузлів для передачі даних у мережі.

Задля уникання затримки в побудові нового маршруту передачі пакетів, при реконфігурації усі параметри якості обслуговування, що були обчислені при функціонуванні мережі раніше, вважаються тими самими. Таким чином, граф із вагами ребер, на основі яких визначаються вартості маршрутів, вже готовий для аналізу контролером. Єдиною технічною задачею, яку йому потрібно вирішити для побудови нового каналу зв'язку – обрати через одноразове задіяння алгоритму Дейкстри найменш вартісний маршрут між вихідною вершиною та адресатом. Опісля того, як стабільний зв'язок буде встановлено, а передача даних відновиться, контролер після проходження встановленого адміністратором періоду часу обчислить параметри якості обслуговування зв'язків та надійність вузлів вже на основі функціонування нових каналів зв'язку. Після цього ним буде проведена

планова реконфігурація топології мережі та, за необхідності, зміна робочих каналів зв'язку.

Топологічний граф мережі, на базі якого контролер проводить реконфігурацію після відключення частини вузлів, зображений на рис. 3.15.

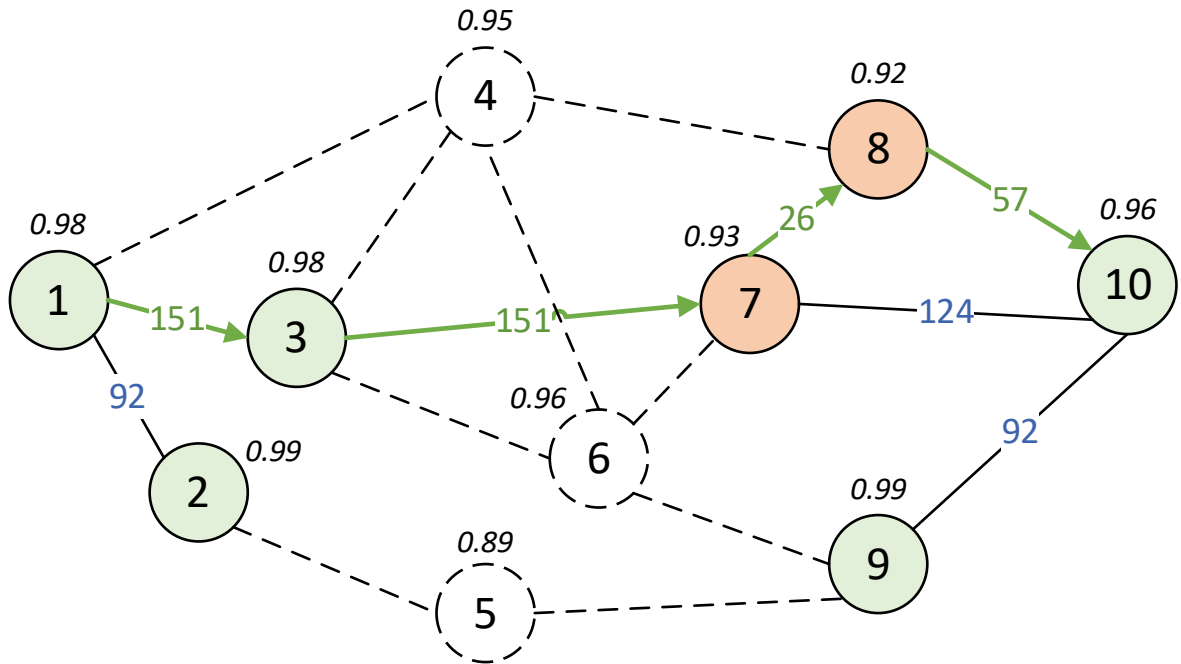


Рис. 3.15. Побудова нового каналу зв'язку після відключення частини вузлів

Замість ділянки  $1 \rightarrow 4 \rightarrow 6 \rightarrow 7$ , що після відключення вузлів виявилася недоступною, для контролера є можливість обрати вартісну ділянку  $1 \rightarrow 3 \rightarrow 7$ , і довести трафік до вершини 10 через робочий відрізок  $7 \rightarrow 8 \rightarrow 10$ , який був присутній у оригінальному маршруті, замість прямого зв'язку  $7 \rightarrow 10$ . У правильності вибору відрізка легко переконатися, оскільки

$$l_{7,8} + l_{8,10} < l_{7,10}, \quad (3.4)$$

де  $l_{ij}$  – вартість маршруту згідно параметрів якості обслуговування.

Отже, за допомогою запропонованого методу динамічної реконфігурації з використанням відмовостійких резервних каналів зв'язку контролер *SDN* має змогу забезпечити безперебійний сеанс передачі трафіку у разі відмови основного каналу зв'язку.

### Висновки до розділу 3

У розділі представлений алгоритм побудови множини каналів зв'язку, що не перетинаються, між вузлами мережі, що є реалізацією одного з етапів відповідного методу побудови каналів зв'язку на основі інтегрального критерію з урахуванням якості обслуговування, запропонованого у розділі 2. Для реалізації алгоритму необхідно обчислити метрики якості обслуговування для кожного із зв'язків між пристроями в мережі. На основі цих даних, зібраних у матрицю, за методом Дейкстри необхідно побудувати множину шляхів, що не перетинаються. Побудовані шляхи, що базуються на даних контролера *SDN* про поточний стан мережі, виступають перерізом для поділу графа мережі. За рахунок використання запропонованого алгоритму і послідовного розділення графу мережі на підграфи, кожний шлях будується зі зменшеною часовою складністю.

У розділі представлений алгоритм відбору каналів зв'язку по критерію надійності вузлів, що є реалізацією наступного етапу методу побудови каналів зв'язку, запропонованого у розділі 2. Використання алгоритму дозволяє збільшити відмовостійкість комп'ютерної мережі та забезпечити гарантію доставки необхідної інформації по безпечним маршрутам. Використовуючи числові коефіцієнти надійності як обмеження для вибору шляху між вершинами, необхідно обрати основний маршрут з множини альтернатив згідно встановленій вимозі до надійності вузлів, інші маршрути вважати резервними. Вимоги до надійності залежать від типу трафіку, що передається у мережі.

У розділі запропоновано удосконалення методу динамічної реконфігурації віртуальної мережі *SDN* з використанням відмовостійких резервних каналів зв'язку. Використання методу дозволяє підтримувати сеанс передачі трафіку у разі відмови основного каналу зв'язку та оперативно обрати альтернативний шлях для надсилання трафіку. Якщо у процесі передачі трафіку виникає збій у проміжних ланках мережі або каналах зв'язку між ними, то для реконфігурації та обходу проблемної ділянки використовуються моніторингові дані *SDN* контролера.

## РОЗДІЛ 4

### РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОБУДОВИ ВІРТУАЛЬНОЇ МЕРЕЖІ НА ОСНОВІ ТЕХНОЛОГІЇ SDN

#### 4.1. Аналіз технічних можливостей протоколу OpenFlow для побудови віртуальної SDN мережі

З-поміж інших рішень протокол *OpenFlow*, який відповідає за комунікацію між *SDN*-контролером та проміжними пристроями в мережі (найчастіше це так звані комутатори *Open vSwitch*), поступово стає стандартом завдяки доступності та достатньо простій реалізації. В рамках пошуку технологій для побудови віртуальної мережі на основі технології *SDN* протокол, що є ланкою для зв'язку між комутаторами та контролером, є фундаментом для побудови усієї інфраструктури, тому обрання правильного рішення впливатиме на показники, яких бажано досягнути при моделюванні. Необхідно провести теоретичний огляд протоколу та проаналізувати його відповідність рішення поставленої задачі побудови програмно- визначеної мережі з більшою швидкістю передачі трафіку та реконфігурації.

*OpenFlow* є відкритим та безкоштовним для впровадження протоколом, що створений для *SDN*-комутаторів, які підтримують віддалене управління, що є основою концепції програмно-визначеної мережі. Втім, для правильної роботи протоколу пристрої у мережі мають бути спеціально апаратно спроектовані для його підтримки [74].

Протокол *OpenFlow* повністю реалізує технічні вимоги, які існують для технології *SDN*. Це включає в себе створення стандартного мережевого обладнання, незалежного від його виробника, оскільки, як зазначалось у минулих пунктах, кожна компанія, що виробляє мережеві пристрої, зазвичай конфігурує своє обладнання власними протоколами, часто під пропрієтарною ліцензією, що робить рішення побудови інфраструктури дорожчим. Крім того, реалізація



*OpenFlow* передбачає комунікацію між контролером та комутаторами через *API*, для чого на контролері запускаються відповідні драйвери. Це дозволяє йому за допомогою директив по протоколу керувати мережею та визначати, які користувачі або підсистеми можуть взаємодіяти між собою [75].

Протокол стає посередником між центральним контролером та проміжними мережевими пристроями. Всі *Open vSwitch*-комутатори мають у своїй пам'яті принаймні одну таблицю потоку (*flow table*) (у версії *OpenFlow 1.0* не було можливості мати кілька таких таблиць) і так званий канал захищеного зв'язку (*secure channel*), через який відбувається комунікація з контролером (рис. 4.1) [76].

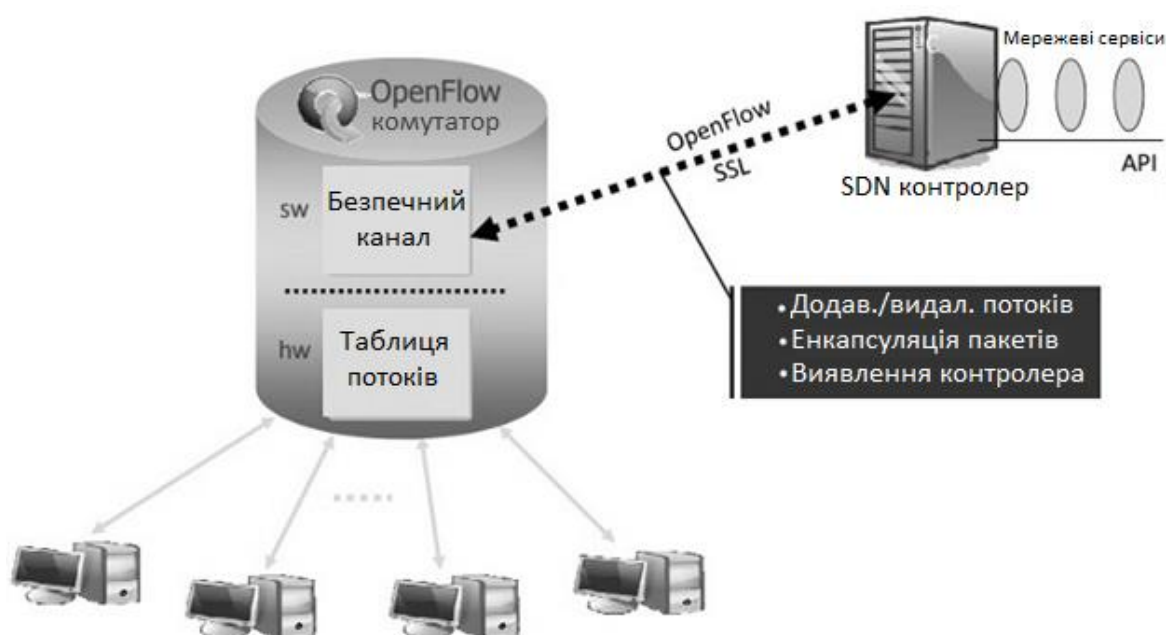


Рис. 4.1. Схема комунікації *SDN*-контролера і комутатора згідно протоколу *OpenFlow*

Таблиця потоків виступає основою для механізму передачі даних між інтерфейсами комутатора. Кожна таблиця містить записи правил (*flow entries*), що в свою чергу містять наступну інформацію:

- поля відповідності (*match fields*) або заголовки слугують ідентифікаторами правил для збереження принципу унікальності;
- лічильники (*counters*) зберігають кількість застосувань правил контролером протягом певного періоду часу та використовуються для збору статистичних даних;

- інструкції (*instructions*) зберігають сутність власне правила [77].

Незважаючи на те, що для підтримки *OpenFlow* пристрій має бути відповідно сконструйований, сам протокол не встановлює конкретний вигляд *API* для своєї роботи. Таким чином, адміністратори мережі мають свободу вибору інтерфейсів та застосунків, які будуть інсталиюватись на контролер.

Коли комутатор, сконфігурований під роботу з протоколом *OpenFlow*, отримує пакет даних від іншого пристрою у мережі, на ньому відбуваються такі операції:

- Комутатор виділяє та запам'ятовує заголовок пакету, який представляє собою послідовність бітів;
- Комутатор розпочинає пошук запису серед доступних, де поле відповідності найбільше збігається із заголовком пакету;
  - Співпадіння знайдено: комутатор виконує інструкції, вказані в цьому записі для цього пакету (наприклад, відправка на інший порт, зміна заголовка, обробка в таблиці, тощо);
  - Співпадіння не знайдено: комутатор або видаляє пакет, або відправляє його до контролера в інкапсульованому вигляді. Потім контролер проводить аналіз та створює правило для обробки цього та подібних пакетів, висилаючи його на комутатор, де воно додається в таблицю потоків [78].

Якщо комутатор з підтримкою *OpenFlow* містить кілька таблиць потоків, то для повного аналізу пакета використовується конвеєр. Починаючи з обробки заголовка пакета в одній таблиці, він потім може бути переданий за допомогою метаданих та проаналізований в інших таблицях. Для цього, записи в таблицях потоків мусять містити конкретні інструкції для конвеєрів.

Найчастішим правилом, яке завантажується до комутатора, є інструкції для пересилання пакета на інший порт. Він не обов'язково має бути фізичним, і для віртуальних мереж можна використовувати віртуальні порти. Якщо порт віртуальний, його можна налаштувати для автоматичного виконання певних дій

без задання додаткових інструкцій – наприклад, відправка даних до контролера, відправка на інший комутатор, перенаправлення без використання *OpenFlow* [79].

Таблиці потоків можуть використовуватись більш гнучко. Вони мають можливість зберігати посилання на набори інструкцій, за рахунок чого інформацію про необхідні щодо даних дії можна інкапсулювати. Серед запитів складнішої семантики можуть бути масова розсилка (*broadcast*), динамічна зміна маршрутів, агрегування каналів. Цей механізм дозволяє змінювати вихідні пакети для великих потоків даних, зберігаючи при цьому ресурси. Групи, подібно до записів, зберігаються в своїх власних таблицях [80].

*OpenFlow* не вимагає конкретної програмної реалізації мережевих пристроїв, але для роботи протоколу потрібна стандартна процедура обробки пакетів, а семантика інструкцій має відповідати конкретним вимогам. Деталі реалізації, такі як використання апаратних таблиць маршрутизації, спільних (*shared*) бітових масок для всіх груп або потоків, можуть варіюватись. Важливо, щоб будь-які дії щодо правил – їхня установка, оновлення, видалення – контролювалися безпосередньо контролером. Він може робити це реактивно (встановлюючи правило після отримання пакету, для якого немає інструкцій) або проактивно (встановлюючи правило перед надходженням пакетів, яким це правило буде потрібне).

*OpenFlow* управляє даними за допомогою системи потоків, завдяки чому відповідні таблиці на комутаторах і отримали свою назву. Це означає, що відповідне правило відпрацьовує не для кожного пакету окремо, а і для наступних з потоку. Такий підхід економить апаратні ресурси для обробки даних, дозволяючи узагальнити процес передачі даних через пристрої.

Комунікація між контролером і комутатором у протоколі *OpenFlow* здійснюється через три види повідомлень:

- прямі – передаються від контролера до конкретного комутатора в мережі і служать для відстеження статусу комутатора, надання інструкцій керування, збирання статистики, редагування записів в таблицях потоків комутаторів;

- симетричні – передаються як від контролера до комутатора, так і навпаки, використовуються для виявлення можливих збоїв і затримок передачі даних;
- асинхронні – передаються від комутатора до контролера для сповіщення його про проходження пакетів, зміни записів в таблицях, зміни стану контролера або виникнення помилок [81].

Отже, протокол *OpenFlow* є доволі простим у реалізації та підтримці засобом забезпечення швидкого зв'язку в мережах *SDN*, завдяки чому вже став своєрідним стандартом для програмно-визначених мереж. Він використовується і у рішенні, що запропоновано в даній роботі.

#### **4.2. Обґрунтування вибору базового програмного забезпечення SDN-контролера**

Існує багато відкритих та комерційних *SDN* контролерів, що підходять для різних застосувань. Контролери в основному поділяються на розподілені та централізовані. Кількість контролерів безпосередньо впливає на продуктивність мереж *SDN* та дотримання якості обслуговування. Проблема розміщення контролерів може впливати на інші аспекти, такі як затримка комунікації між контролерами та комунікація між контролером і комутаторами у мережі.

Контролери *Floodlight* базуються на *Java* та підтримують як віртуальні, так і фізичні *OpenFlow*-комутатори. Контролер *Floodlight* реалізує модуль, який забезпечує такі функції, як видалення потоків, вставка потоків та деякі політики для управління *QoS*. Ці модулі реалізуються в протоколі *OpenFlow* версії 1.0, який здатен відстежувати політики в комутаторах та застосовувати їх до класів сервісів. У північному інтерфейсі *Floodlight* модуль *QueuePusher* генерує повідомлення для конфігурації черг для створення, читання, оновлення та видалення функцій для управління комутаторами типу *Open vSwitch* [82]. Втім, використання лише протоколу *OpenFlow* для управління потоками дещо обмежує цей контролер у його можливостях керування мережею та забезпечення якості обслуговування.

Відкритий контролер *OpenDaylight (ODL)*, що базується на *Java*, реалізує інфраструктуру *SDN*. *OpenDaylight* підтримує програмування за допомогою фреймворків *OSGi* для локально розташованих контролерів та програмування за допомогою *REST* для віддалених контролерів. Платформа *ODL* розробляє деякі додатки (наприклад, координатори віртуалізації та захист від розподілених *DoS*-атак) та кілька плагінів протоколів південного інтерфейсу для гетерогенних мереж. Для досягнення *QoS* в мережах на основі потоків *ODL-Lithium* може бути застосований плагін південного інтерфейсу для інфраструктури *DOCSIS*. Інший плагін південного інтерфейсу *OVSDb* конфігурує та керує чергами в комутаторах *SDN*. Крім того, для *QoS* в цьому контролері є додаток *Packet Cable Multimedia (PCMM)*, який забезпечує інтерфейс для управління поточковими сервісами для мереж *CMTS*. Додатково в модулі резервування *ODL* передбачено надання низькорівневого резервування ресурсів, яке забезпечує мережеві послуги, пропускну здатність, порти для користувачів на певний виділений час [83].

Контролер *ONOS* базується на *Java* та підтримує як віртуальні, так і фізичні комутатори *OpenFlow*. Він забезпечує розподілені платформи, що покращують продуктивність, масштабованість і доступність мереж для постачальників послуг. Однак підтримка механізму *OpenFlow* обмежує використання *ONOS* лише комутаторами, що підтримують цей протокол. Для покращення *QoS* в бібліотеках *ONOS* реалізовано клас *SetQueueInstruction* (високорівнева інструкція). З параметрів якості обслуговування мережі, які можна покращити за допомогою цього контролеру, слід виділити надійність, масштабованість, узгодженість та балансування навантаження [84]. Важливими перевагами використання *ONOS* є простота реалізації взаємодії контролера з комутаторами (через *REST API*), а також значна кількість додатків у відкритому доступі, які дозволяють значно розширити можливості контролера.

У розподілених мережах через архітектурні проблеми контролер може вводити несумісні правила потоків у комутатори, що потенційно призводить до нестабільності мережі *SDN*. Ця нестабільність може спричинити переривання передачі даних, втрату пакетів та асинхронну комунікацію між контролерами.

*DevoFlow* та *DIFANE* вирішують ці проблеми, впроваджуючи контрольну метрику в комутаторах, що, однак, суперечить принципам дизайну *SDN*. Для підтримки узгодженості були запропоновані рішення, такі як адаптивна модель узгодженості, техніки кластеризації, самонавчальні моделі узгодженості. Однак проблеми узгодженості між контролерами усе ще не мають готового рішення [85].

Розрив з'єднання між рівнем даних і рівнем управління через відмову може спричинити збій мережі. Тому питання надійності при задіянні кількох контролерів є вкрай важливим. Для управління проблемами відмови серед контролерів використовуються метрики мінімального розрізу контролерів та евристичні метрики, які обчислюють надійність контролера в різних топологіях. Однак через високу вартість більшості контролерів єдиної ефективної метрики надійності наразі не знайдено.

У середовищі розподілених контролерів кожен комутатор адмініструється контролером у локальному домені. Службовий трафік управляється контролерами, і зі збільшенням кількості правил потоків деякі контролери стають перевантаженими, а деякі – залишаються недовантаженими. Такий дисбаланс погіршує продуктивність мережі *SDN* через низьку пропускну здатність та високий час відповіді. Для зменшення навантаження на контролери використовується модуль переназначення, який моніторить і збирає статистику використання контролерів, а також переназначає асоціації між контролером і комутатором. У деяких випадках неактивний контролер може бути видалений, і буде додано новий контролер для управління навантаженням та ефективністю.

На основі проведеного аналізу контролерів *SDN* для виконання практичної частини дослідження було обрано контролер *ONOS*. Цей контролер розповсюджується по відкритій моделі, що дозволяє отримати доступ до його коду та модифікувати його на будь-якому рівні. *ONOS* виставляє порівняно невеликі вимоги до апаратного забезпечення, тому може бути запущений на персональному комп'ютері. Перевагою цього контролера також є велика база різноманітних додатків, які дозволяють значно розширити можливості контролера по моніторингу та передачі даних у мережі *SDN*.

### 4.3. Програмна реалізація обчислення мінімального значення метрики якості обслуговування каналу зв'язку

Першим етапом процесу побудови каналів зв'язку та динамічної реконфігурації є створення множини шляхів згідно алгоритму, схематично зображеного на рис. 3.2. Для реалізації пошуку мінімальної ваги ребра програмним чином, необхідно представити цей процес у вигляді програмного коду. Ця логіка буде використовуватись у обчисленнях SDN-контролера в ході виконання ним задачі побудови віртуальної мережі та каналів зв'язку. Оскільки у пункті 4.2 було обґрунтоване використання контролера *ONOS*, що написаний на мові *Java*, представлений нижче код використовує оператори цієї мови (рис. 4.2).

```
int getMin() {
    if (dist.length() < 1) {
        return minStruct { index = -1; val = 0; }
    }
    int i,minIndex = 0;
    int minVal = MAX_NUM;

    while (i < len(dist)) {
        if (v_in_S(i + 1) == false) {
            if (dist[i]<MAX_NUM && dist[i]>0 && minVal>dist[i]) {
                minVal = dist[i]; minIndex = i;
            }
        }
        inc(i);
    }

    return minStruct { minIndex + 1, minVal };
}
```

Рис. 4.2. *Java*-код для функції пошуку мінімальної ваги ребра алгоритмом Дейкстри для програмного забезпечення *SDN*-контролера

Використовуючи функцію *getMin*, описану вище, можна порівняти значення *dist* (дистанція) вузла для наступних зв'язаних з ним вершин. Циклічна перевірка робиться для кожного вузла, і у разі меншого значення *dist* для вузла, що

перевіряється, порівняно з попереднім, нинішній вузол обирається як направлення загального шляху. Значення лічильника  $i$  збільшується на одиницю після проходження циклу для кожної вершини. Цей лічильник потрібен, щоб виконати цикл необхідну кількість разів та завершити аналіз усіх вершин.

Значення  $dist$  в ході циклу оновлюється після того, як для кожного наступного (за нинішнім) вузла виконується мінімальне порівняння з попередньо проставленим значенням відстані. Остаточне значення  $dist$  використовується, коли точно визначений наступний вузол для маршруту. Під час роботи програми змінна  $dist$  є локальною для циклу, оскільки для роботи алгоритму пошуку шляху необхідне лише її остаточне значення.

При врахуванні параметрів якості обслуговування, значення  $dist$  буде представляти собою поєднане значення функцій параметрів  $QoS$ , які згідно формули 2.12 сумуються в умовну вагову матрицю, яку здатен використовувати алгоритм Дейкстри та його програмна реалізація. Алгоритм шляхом побудови найкоротшого шляху вирішує задачу створення каналу зв'язку, що відповідає параметрам  $QoS$ .

Наступним етапом побудови шляхів між вершинами є вилучення каналів зв'язку, що проходять через проміжні вершини, які не відповідають критеріям надійності, встановленим згідно вимог до мережі. Метрика надійності вузла є відношенням пакетів, що успішно вийшли через пристрій від вихідного порту та тими, що надійшли до вхідного порту. Відповідно, величина надійності кожного вузла має виражатися у дробовому значенні від 0 до 1, а обчислення проводиться на основі моніторингу передачі даних по мережі.

Після встановлення значень надійності вузлів контролеру необхідно виключити канали зв'язку, що проходять через ненадійні вершини, з множини альтернативних каналів зв'язку згідно алгоритму, зображеного на рис. 3.11. Процес цього виключення у вигляді програмного *Java*-коду зображений на рис. 4.3.



```

boolean checkPathReliability() {
    Set<Vertex> vertices = path.getVertices();
    Map<Long,Float> reliabilityCoefficients =
        graph.getVertices().getCoefficients();
    for (int i = 0; i < vertices.length(); i++) {
        if (vertices[i] < requiredReliability) {
            disablePath(path);
            return false;
        }
    }
    return true;
}

```

Рис. 4.3. *Java*-код функції визначення проходження шляху через комутатори, що не відповідають критеріям надійності

Згідно цього підходу шлях розбивається на масив окремих вершин. Для кожної з цих вершин в графі визначається коефіцієнт надійності. Потім в алгоритмі запускається цикл, в якому значення коефіцієнту порівнюється із потрібною надійністю *requiredReliability*.

Якщо надійність хоча б однієї вершини нижча за потрібну, шлях отримує стан *disabled*, а функція *checkPathReliability* повертає значення *false*. Якщо перевірка надійності пройдена для всіх вершин і значення дорівнює або більше за необхідне, шлях вважається достатньо надійним, і функція повертає значення *true*.

В процесі передачі трафіку може виникнути необхідність реконфігурації мережі у зв'язку з відключенням частини вузлів. У цьому випадку контролер приймає відповідне повідомлення, після чого перенаправляє трафік по резервному маршруту, і паралельно формує нові на основі вузлів, що залишились у робочому стані. У програмному вигляді це реалізується через виконання відповідної функції, що модифікує набір альтернативних шляхів, щоб залишити лише ті, які проходять через вузли, що зберегли здатність до передачі даних. Функція викликається після надходження на контролер повідомлення про відключення вузлів, а у вигляді програмного *Java*-коду її можна зобразити наступним чином (рис. 4.4).

```

void reconfigure(Set<Vertice> failedVertices) {
    foreach (path : paths) {
        if (path.getVertices().contains(failedVertices)) {
            disablePath(path);
            paths.removePath(path);
        }
    }
    if(!paths.isEmpty()) {
        setOptimalPath(paths.getOptimal());
    } else {
        Matrix updMatrix = updateWeightMatrix(graph);
        Set<Path> updPaths = formDisjointPaths(updMatrix);
        setOptimalPath(paths.getOptimal());
    }
}

```

Рис. 4.4. *Java*-код функції динамічної реконфігурації мережі

Функція *reconfigure* приймає у якості параметрів набір вершин, які зазнали збою. Потім для кожного шляху з множини перевіряється, чи містить даний шлях хоча б одну з вершин, що перестали працювати. Якщо містить, тоді шлях вимикається функцією *disablePath*, після чого видаляється з набору шляхів методом *removePath*.

Пройшовши через усі шляхи та виключивши проблемні, функція перевіряє, чи залишилися у наборі активні шляхи. Якщо такі ще є, вона використовує метод *getOptimalPath*, що назначає основним маршрутом передачі даних шлях із найменшим значенням довжини з набору. Якщо ж після видалення проблемних шляхів набір повністю вичерпався, функція переходить до реконфігурації та перестворення набору шляхів, що не перетинаються.

Вона створює нову матрицю ваг графа, оновлюючи її функцією *updateWeightMatrix*, враховуючи втрату частини вершин. Ця матриця представляє граф, де проблемні вершини відсутні. З цієї оновленої матриці ваг функція формує новий набір шляхів функцією *formDisjointPaths*. Після цього, з нового набору шляхів за допомогою функції *setOptimalPath* вибирається новий шлях та назначається як найпріоритетніший для передачі трафіку.

Отже, ця функція, з одного боку, намагається знайти найкращий можливий шлях, використовуючи діючі вершини і шляхи, що вже були раніше знайдені

контролером, а з іншого – динамічно адаптується під зміни у графі, знаходячи нові шляхи в разі відмови вершин, виконуючи відповідно динамічну реконфігурацію.

#### 4.4. Конфігурація SDN-контролера для проведення моделювання мережі

Для моделювання віртуальної мережі була використана програма *mininet*, модель якої *SDN* контролер відтворює в своїй пам'яті. Утиліта *mininet* дозволяє побудувати топологію мережі на основі *python* скрипту, в якому визначена інформація про ряд мережевих параметрів: характеристики і опис пристроїв, властивості зв'язків між ними, версії протоколів маршрутизації, і так далі. Для більшої зручності та економії часу при проектуванні мережі, було додатково використане розширення *mininet* під назвою *miniedit*, яке дозволяє будувати топологію мережі за допомогою графічного інтерфейсу (рис. 4.5).

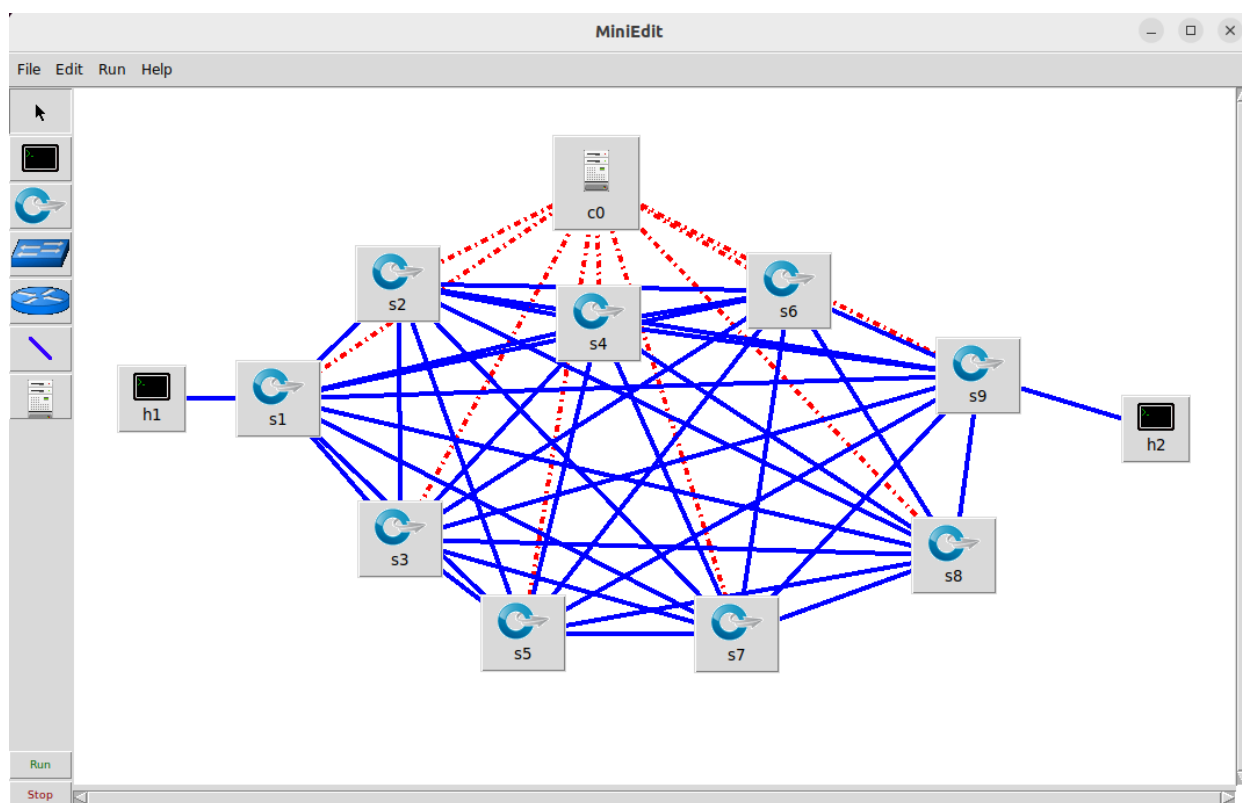


Рис. 4.5. Приклад побудови повнозв'язної віртуальної мережі в утиліті *miniedit*

Для проведення процесу тестування створена модель, котра має наступні особливості:

- два хости (комп'ютери користувачів), що об'єднані між собою у мережу з адресою 10.0.0.0/8, саме між цими хостами необхідно побудувати канал зв'язку для передачі трафіку;
- комутатори *Open vSwitch*, що встановлюють між собою зв'язок та передають трафік на базі протоколу *OpenFlow*;
- *SDN* контролер, що виконує адміністрування та реконфігурацію усіх комутаторів в мережі;
- працює на версії протоколу *OpenFlow* 1.3 для коректної сумісності з програмним забезпеченням контролера *ONOS*.

Під час запуску моделювання в програмі була налаштована можливість запуску командного інтерфейсу користувача (*CLI*), що забезпечує введення і виконання необхідних команд під час активного використання моделі. Ця особливість дозволяє користувачам вводити, модифікувати та виконувати команди в реальному часі безпосередньо під час процесу моделювання. Це спрощує роботу з мережею, дозволяючи легко контролювати процеси відлагодження та тестування (рис. 4.6) [86].

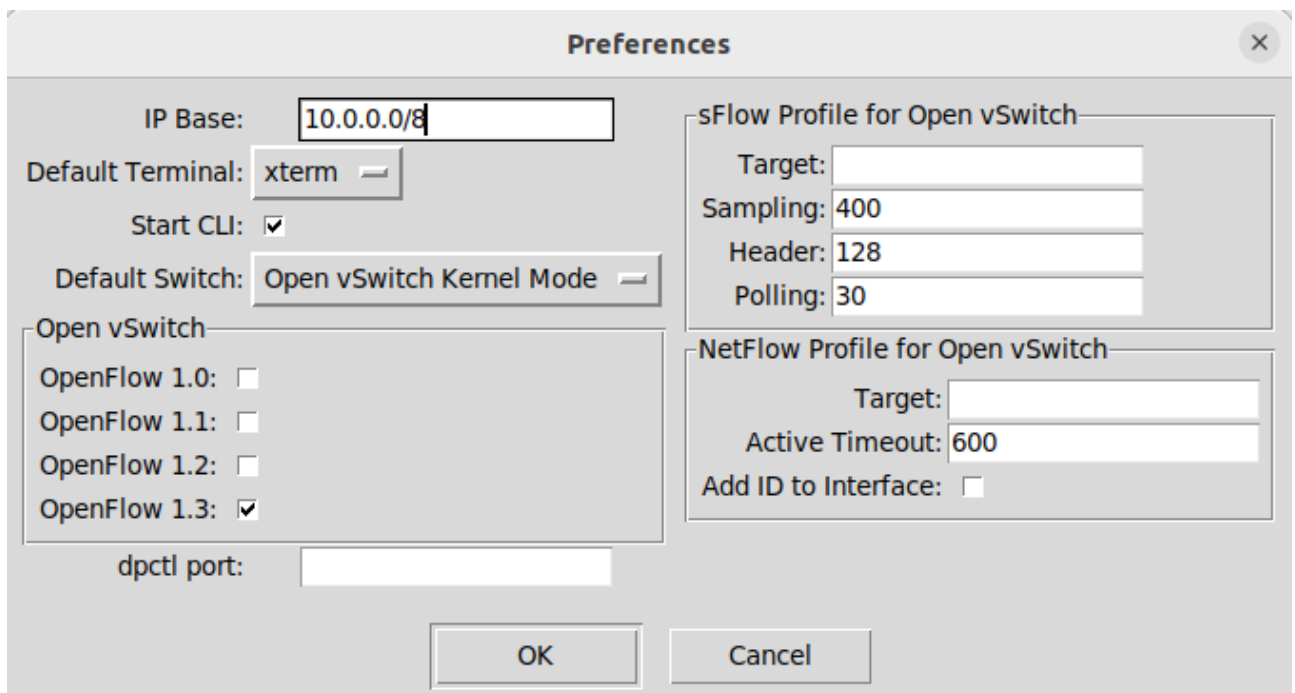


Рис. 4.6. Налаштування віртуальної мережі *mininet* (використовуючи *miniedit*)

Апаратним забезпеченням для *SDN* контролера є комп'ютер, що працює під управлінням *Ubuntu Linux*. Від адміністратора потребується лише встановлення програмного забезпечення *ONOS*.

*ONOS* є відкритим (*open source*) програмним забезпеченням для контролера і позиціонується як рішення для мережі, що складається з постачальника послуг зв'язку та його клієнтів. *ONOS* є зв'язуючою ланкою для більш простих *SDN*-систем, будучи своєрідним кластером для декількох вузлів. Перевагою використання *ONOS* у якості системи для контролера мережі є можливість цього програмного забезпечення протидіяти збоєм системи у разі порушення налаштування окремих вузлів. Окрім цього, вузол, у якому сталася помилка адресації або з'єднання, може бути оперативно переконфігурований програмним забезпеченням контролеру.

Вважати, що *OpenFlow* та інші подібні мережеві протоколи інтегровані в *ONOS*, буде некоректно, оскільки *OpenFlow* використовується лише для встановлення з'єднання між пристроями мережі, а сама операційна система *ONOS* забезпечена власними додатками та програмними рішеннями. Вони розділені на рівні (*tiers*), на кожному з яких працюють власні програмні моделі (одна з них взаємодіє з *OpenFlow*) і які можуть функціонувати незалежно. Завдяки подібній ізоляції рівнів система набуває універсальності, оскільки не прив'язується до однієї моделі або протоколу і може взаємодіяти на різних рівнях з іншими елементами мережі в межах топології.

На базі рівнів та моделей у *ONOS* працюють додатки (*applications*), за допомогою яких програміст має можливість налаштовувати мережу та контролювати трафік. У пристрої управління, на якому працює операційна система, зберігається інформація про топологію мережі. З цього ж центру є можливість встановити зв'язок безпосередньо з певним пристроєм у мережі та надіслати йому певні дані. Це розширює можливості контролювання мережі та спрощує розробку додатків та адміністрування. Власне додатки можуть бути додані або вилючені з мережі динамічно, для цього не потрібно зупиняти трафік та відключати або перезапускати мережу, що є величезною перевагою для

провайдерів, які потребують безперервного з'єднання з абонентами [87].

Для експерименту, що проводиться у цьому розділі, важливою є відкритість програмного забезпечення контролеру *ONOS*, оскільки це дозволяє не збільшувати матеріальні витрати для побудови мережевого рішення та відкриває можливість необмеженого внесення змін до коду додатків контролера.

Як уже зазначалося вище, на *ONOS* можна включати існуючі та додавати нові додатки для розширення функціоналу контролера. Для експерименту використовуються декілька додатків, які включені в стандартний пакет установки розробниками *ONOS*, функціональний код цих додатків був модифікований в ході дослідження:

- *OpenFlow Provider Suite* (*org.onosproject.openflow*) – застосунок, що дозволяє контролеру працювати з протоколом *OpenFlow*, взаємодіючи з *openvSwitch* комутаторами через спеціалізований програмний інтерфейс (*API*);
- *Reactive Forwarding* (*org.onosproject.fwd*) – застосунок, що дозволяє контролеру виконувати маршрутизацію трафіку за рахунок стандартних директив та без можливості урахування *QoS*; використовувався для порівняння результатів моделювання;
- *Intent Forwarding* (*org.onosproject.ifwd*) – модифікований в ході дослідження застосунок, що дозволяє контролеру виконувати маршрутизацію трафіку за рахунок встановлення спеціальних правил – намірів (*intents*);
- *LLDP Link Provider* (*org.onosproject.lldpprovider*) – застосунок, що забезпечує можливість для ядра *ONOS* виявляти вузли топології та зв'язки між ними шляхом відслідковування контрольних пакетів протоколу *LLDP*;
- *Proxy ARP/NDP* (*org.onosproject.proxyarp*) – застосунок, що дозволяє контролеру направляти *ARP/ICMP* трафік, що дає можливість перевіряти зв'язок між хостами у мережі.

Після запуску *ONOS* на комп'ютері та старту моделювання у *mininet* дві утиліти встановлюють зв'язок одна з одною, а машина, де запущено *ONOS*,

представляється у топології як контролер *c0*. На контролері будується власне представлення топології мережевої інфраструктури. Її можна побачити, підключившись до системи *ONOS*, використовуючи графічний інтерфейс, що входить в пакет програмного забезпечення.

В ході дослідження топології контролер визначає кількість комутаторів та хостів, що поєднані між собою зв'язками (*links*). Для функціонування передачі даних у *SDN* мережі використовуються потоки (*flows*). Вони являють собою дані, що додаються контролером до конфігурації кожного комутатора та дозволяють останньому робити обробку трафіку. Кожен запис потоку має поле *selector*, яке визначає тип трафіку, що надходить на комутатор. Ці потоки є своєрідною реалізацією таблиць маршрутизації у протоколі *OpenFlow*.

#### 4.5. Порядок проведення моделювання

Задачею у експерименті є побудова каналу зв'язку та передача даних між двома хостами – *h1* (IP-адреса: 10.0.0.1) та *h2* (IP-адреса: 10.0.0.2). Щоб дістатись від першого хоста до другого, трафіку потрібно пройти через декілька *Open vSwitch* комутаторів. Якщо усі комутатори та зв'язки між ними знаходяться у робочому стані, то будуть побудовані шляхи, що не перетинаються, які будуть використовуватися для створення каналів зв'язку [86].

Після опізнавання контролером топологічної організації віртуальної мережі спочатку вона відображається без хостів *h1-h2*, визначених на етапі проектування. Це пов'язане з тим, що службовий трафік, який відслідковує додаток *LLDP Link Provider* і який потім створює у пам'яті контролера представлення про мережу, проходить лише між *Open vSwitch* комутаторами, а до хостів не відправляється. Для їхнього відображення необхідно виконати перевірку зв'язку між усіма хостами через відправку тестового *ICMP* трафіку за допомогою команди *pingall*. Це дозволить перевірити не лише доступність хостів один для одного, але й дасть можливість контролеру провести дослідження маршрутів у мережі для подальшого запам'ятовування і використання.

Втім, обрана реалізація пошуку шляхів зв'язку у контролері *ONOS* потребує додаткової дії. Щоб успішно встановити зв'язок від одного хоста до іншого, між ними треба створити спеціальний запит – намір (*intent*). Намір – це об'єкт у абстрактному представленні програмно-визначеної мережі, який описує запит програми до ядра *ONOS* для зміни поведінки мережі. Наміри, зокрема, описують наступні елементи поведінки:

- Мережевий ресурс: набір об'єктних моделей, таких як зв'язки, які сформовані в межах частин мережі, на які впливає намір;
- Обмеження: вагові коефіцієнти, застосовані до набору мережевих ресурсів, таких як пропускна здатність, оптична частота або тип зв'язку;
- Критерії: поля заголовка пакета або шаблони, які описують фрагмент трафіку;
- Інструкції: дії з адміністрування фрагмента трафіку, як-от, модифікація поля заголовка або виведення фрагменту через певні порти.

Крім того, намір завжди ідентифікується як через *ApplicationId* програми, яка додала його в систему, так і унікальним *IntentId*, згенерованим під час створення наміру [84].

Однією з головних переваг використання намірів перед записами потоків для програмної конфігурації мережі є те, що наміри відстежують стан мережі та динамічно змінюють свою конфігурацію, щоб задовольнити поставлену перед наміром задачу пересилання трафіку. Наприклад, якщо певні зв'язки між вузлами стануть недоступними, то намір перенаправить потік трафіку на альтернативний шлях. Побудова множини альтернативних каналів зв'язку відбувається на основі пріоритетності, яка в свою чергу базується на надійності вузлів та якості обслуговування так, як це було описано у розділах 2 і 3 даної роботи. Якщо альтернативного шляху немає, то намір увійде в невдалий стан (*failed*) і залишатиметься у ньому, доки не стане доступним хоча б один шлях між заданими хостами.

До контролера *ONOS* можна підключитись з використанням командного рядка (*CLI*). Це дає можливість вводити команди для додавання або видалення



сутностей у програмній моделі мережі у пам'яті контролера, а також статистичну інформацію.

Наміри бувають кількох типів, як-от, намір між хостами, точками, колекція зв'язків, намір мережі *MPLS* та інші. Усі наміри за своєю логікою наслідують клас *ConnectivityIntent* у програмній структурі *ONOS*. У рамках моделювання, що проводиться в даному розділі, брались до уваги два типи намірів, що часто використовуються на практиці – намір між хостами (*HostToHostIntent*) та точка до точки (*PointToPointIntent*).

Намір між хостами створюється командою *add-host-intent*, де у якості параметрів вказуються ідентифікатори хостів, між якими необхідно встановити з'єднання та подальшу передачу трафіку. Список намірів з інформацією про їхні налаштування можна отримати за допомогою команди *intents*.

Зазначений вище намір між хостами (*host intent*) є об'єктом *HostToHostIntent* у структурі *ONOS*. Він регулює створення потоків для передачі даних і є абстракцією двонаправленого зв'язку між кінцевими вузлами. Цей намір адмініструє передачу трафіку між хостами і виконує реконфігурацію своїх налаштувань у разі неможливості надсилання трафіку обраними маршрутами. Вибір маршруту *host intent* виконує з урахуванням короткості та завантаженості маршруту між вузлами, однак суттєвим недоліком його застосування є неможливість перенаправлення трафіку по альтернативному маршруту без переривання передачі, навіть у разі якщо дані трафіку дозволено розділити. Крім того, *host*-намір доволі обмежений у кастомізації, через що не може бути пристосований до реалізації поставлених у дисертаційній роботі завдань.

Провести перевірку запропонованих методів побудови віртуальної мережі було вирішено за допомогою більш низькорівневих намірів – точка до точки (*point intent*), які у структурі контролера *ONOS* є об'єктами класу *PointToPointIntent*. Цей тип наміру встановлюється між портами окремого *open vSwitch* комутатора – вузла графу мережі. У кожного такого наміру є набір налаштувань, зокрема пріоритет (*priority*), значення якого можна встановлювати.

В рамках експерименту, що проводиться у даній роботі для перевірки

запропонованих методів побудови множини шляхів, пріоритет наміру точка до точки вважатиметься вартістю переходу, до якого веде вихідний для наміру порт комутатора. Модифікований додаток *Intent Forwarding* на основі метрик якості обслуговування каналів зв'язку та коефіцієнтів пріоритезації для кожного зв'язку між вузлами виконає обчислення вартості маршруту згідно кожної з метрик, зробить сумування вагів та отримає комбіновану вагову матрицю переходів для графу мережі. Після цього для кожного вихідного порту кожного комутатора, що приєднаний до відповідного зв'язку, додаток створить *point intent* та встановить йому пріоритет, що є зворотно рівним (відносно максимального значення ваги зв'язку в умовній матриці) сумарній вартості зв'язку згідно врахованих параметрів якості обслуговування (формула (4.1)).

$$p_x = \max(K_{i,j}) - k_x + 1, \quad (4.1)$$

де  $p_x$  – пріоритет наміру,  $\max(K_{ij})$  – найбільша з вагів ребер у об'єднаній ваговій матриці,  $k_x$  – вага зв'язку.

Контролер *ONOS* у рамках функціональності *point intent*, маючи інформацію про пріоритетність зв'язків, побудує маршрути і на їхній основі канали зв'язку між комутаторами, що безпосередньо підключені до хостів. Основним маршрутом передачі даних буде обраний такий, що має якомога більше значення пріоритетів *point*-намірів. Будуть за можливості побудовані й інші маршрути що не перетинаються між собою. Вони будуть задіяні при передачі даних у разі великого завантаження основного маршруту.

При виході з ладу окремих ділянок мережі додаток *Intent Forwarding* виконує реконфігурацію. Вона полягає, як і слідує з опису у розділах 2 і 3, у виборі одного з альтернативних каналів зв'язку в якості основного, а також побудову нових шляхів з урахуванням відсутності ряду вершин та зв'язків на графі мережі, а також вже функціонуючих відомих маршрутів. На відміну від хост-наміру, де критерієм успішності передачі пакету є його прийняття хостом-адресатом, при використанні *point*-намірів трафік покроково регулюється ними на кожному вузлі. Це дозволить контролеру, що у реальному часі відслідковує зміни у мережі, без зупинки передачі

трафіку перенаправити його іншим маршрутом та знизити часову складність перебудови каналів зв'язку.

У реалізації *ONOS* за замовчуванням всі наміри мають однаковий пріоритет, а побудова каналів зв'язку відбувається лише базуючись на кількості переходів, що існують між джерелом і адресатом. Це призводить до ситуації, коли вже перевантажений найкоротший маршрут може бути обраний для передачі трафіку, у той час як інші маршрути простоюють. Як результат, виникає висока ймовірність втрат даних та збільшення затримки, що погіршує якість обслуговування користувачів мережі. В рамках експерименту порівнюватиметься, зокрема, стандартна реалізація побудови віртуальних каналів зв'язку контролером *ONOS* та модифікована в процесі дослідження, представленого у даній роботі.

#### 4.6. Побудова віртуальної мережі на основі технології SDN

В ході передачі трафіку між хостами контролер відслідковує завантаженість вузлів. Щоб отримати більш репрезентативну картину роботи запропонованого методу реконфігурації, в ході моделювання поступово відключаються найбільш завантажені вершини, аж доки не будуть відключені 50% вузлів мережі. В експерименті порівнюються:

- Стандартна побудова каналів зв'язку у контролері *ONOS* додатком *Reactive Forwarding*, який при побудові каналів зв'язку враховує лише кількість переходів між комутаторами;
- Побудова каналів зв'язку за допомогою *host*-намірів, що враховують лише кількість переходів та ширину пропускання зв'язків, а також не має можливості перенаправити трафік без переривання передачі;
- Побудова каналів зв'язку за допомогою *point*-намірів, що враховують лише кількість переходів та ширину пропускання зв'язків без встановлення пріоритетів;
- Модифікована побудова каналів зв'язку за допомогою *point*-намірів, що враховує чотири метрики якості обслуговування та встановлює відповідні

пріоритети для зв'язків між комутаторами для обходу проблемних ділянок мережі (реалізація методів, що запропоновані у розділах 2 і 3).

Команда *Iperf* з параметрами дозволяє обмінюватися великими об'ємами трафіку між хостами і отримувати статистичні дані щодо цього трафіку. У мережі з  $n$  вершин спочатку починають надсилатися  $n/2$  наборів даних, щоб створити завантаження для зв'язків мережі. Потім протягом 10 секунд іде передача контрольного набору даних. Для цього набору обчислюється кількість переданих даних («трансфер» у термінології утиліти *iperf*) та швидкість передачі («бітрейт» у термінології утиліти *iperf*).

Для урахування метрик якості обслуговування каналів зв'язку було обрано два варіанти конфігурації коефіцієнтів пріоритезації: один для нееластичного трафіку, що потребує високої гарантії доставки, та еластичного (потокowego), для якого допустимі певні втрати даних і затримки. Коефіцієнти були назначені наступним чином:

- нееластичний трафік: кількість переходів – 1, доступність смуги пропускання – 5, затримка – 10, відсоток втрачених пакетів – 10. Коефіцієнт надійності для вузлів – 0,95.
- еластичний трафік: кількість переходів – 5, доступність смуги пропускання – 10, затримка – 4, відсоток втрачених пакетів – 6. Коефіцієнт надійності для вузлів – 0,9.

Для кожного типу топологічної організації проведено три спроби побудови віртуальної мережі. 50% від загального об'єму даних складає еластичний трафік, інші 50% – нееластичний. Результати вимірів швидкості та об'єму переданих даних, представлені далі, є середнім значенням для отриманих при кожній побудові відповідних результатів.

Фактори, що можуть впливати на репрезентативність результатів моделювання:

- модель віртуальної мережі ідеалізована, тому втрати пакетів через недоліки обладнання зведені до нуля;
- зв'язаність топології має бути достатньою, щоби при відключенні окремих

вузлів топологічний граф не був розірваний, через це для тестування не відходить така топологія як дерево (*tree*);

- різниця в швидкості, виміряній відправником й адресатом, пов'язана з різним часовим інтервалом, з яким адресат приймає та оброблює надіслані упродовж 10 секунд (час тесту) пакети – частина з них може залишитись необробленою при настанні моменту кінця тесту.

#### 4.6.1. Повнозв'язна топологія

Повнозв'язний топологічний малюнок є таким, у якому всі вузли мережі зв'язані між собою безпосередньо. При русі з будь-якого комутатора трафік має можливість дістатись до будь-якого іншого за один крок навіть при відключенні окремих вузлів.

Перед основним моделюванням було проведене тестове, де був перевірений розподіл трафіку під час надсилання обмеженого набору зі 100 пакетів між вузлами. Пошук маршруту здійснювався з використанням алгоритма Дейкстри (рис. 4.7).

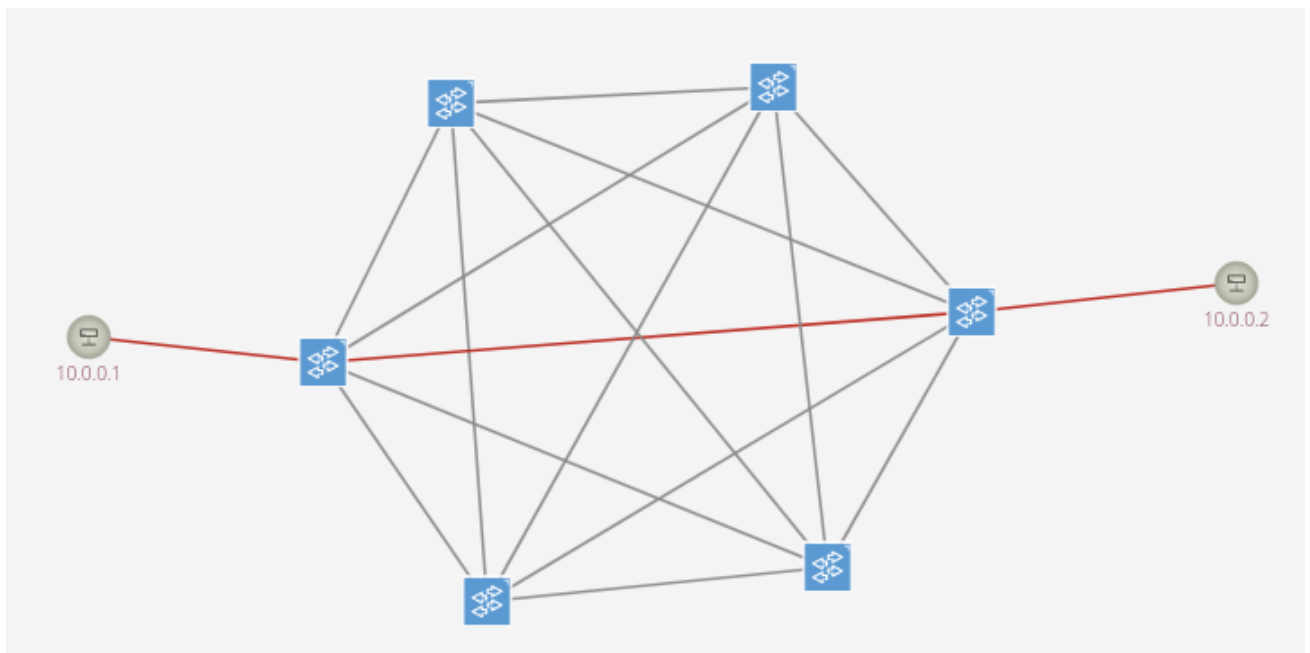


Рис. 4.7. Представлення повнозв'язного графа на контролері *ONOS*

Результати моніторингу показали, що відбулось перевантаження основного

маршруту з 10% втратою пакетів (рис. 4.8) [72].

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 90 received, +10 errors, 10% packet loss, time 101190ms
rtt min/avg/max/mdev = 0.031/0.195/10.661/1.111 ms, pipe 4
```

Рис. 4.8. Статистика доставки пакетів стандартними засобами *ONOS* для повнозв'язного графа мережі

Методи, запропоновані у розділах 2 і 3, дозволяють не допустити цього перевантаження каналу, зокрема і для *SDN* контролера *ONOS*.

Повнозв'язна топологія з дев'ятьма вершинами зображена на рис. 4.9, результати моделювання передачі трафіку для цієї топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.1.

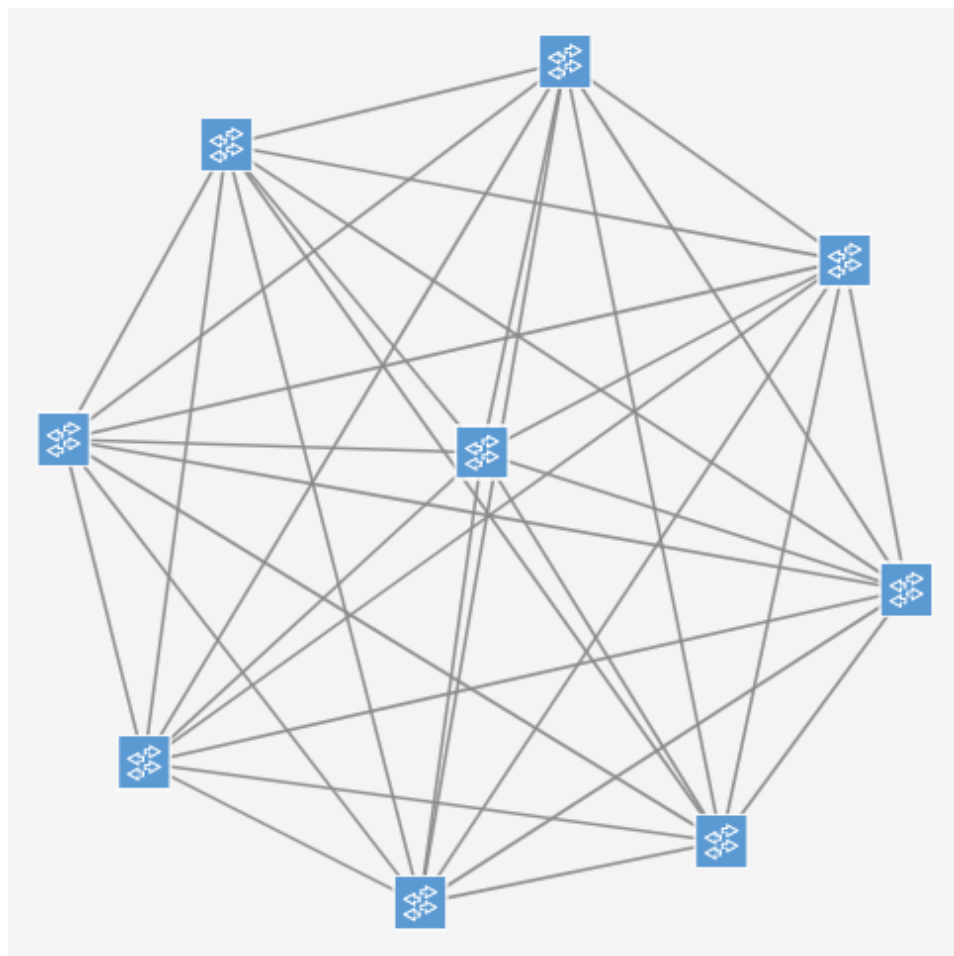


Рис. 4.9. Повнозв'язна топологія (9 вершин) у візуалізації контролера *ONOS*

Таблиця 4.1

Порівняльна статистика об'єму (О) та швидкості (Ш) передачі трафіку для повнозв'язної топології із 9 вершинами

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
9	відправник	49,2	42,3	48,6	41,8	50,4	43,8	52,5	44,8
	адресат		42,1		41,6		43,6		44,7
7	відправник	47,5	40,9	47,1	40,1	48,3	41,4	49,7	41,9
	адресат		40,7		40,0		41,2		41,7
5	відправник	45,8	38,4	43,6	37,5	45,5	38,8	47,1	40,1
	адресат		38,3		37,4		38,7		39,9

Повнозв'язна топологія з двадцятьма п'ятьма вершинами зображена на рис. 4.10.

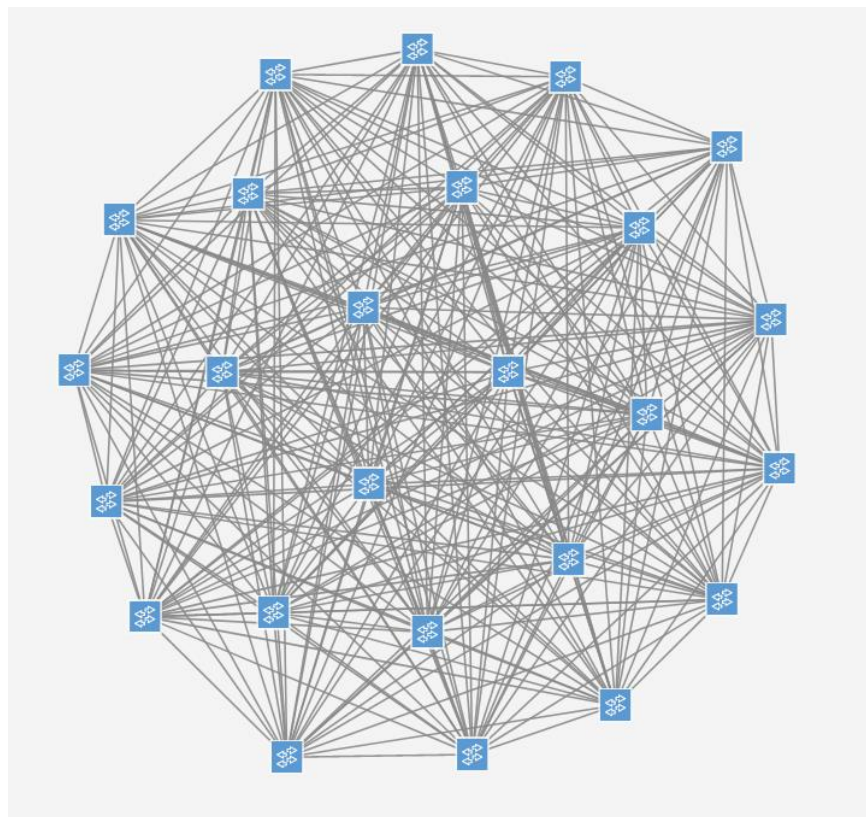


Рис. 4.10. Повнозв'язна топологія (25 вершин) у візуалізації контролера *ONOS*

Результати моделювання передачі трафіку для цієї топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.2.

Таблиця 4.2

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для повнозв'язної топології із 25 вершинами

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
25	відправник	50,8	43,5	51,4	44,5	50,5	43,1	52	44,6
	адресат		43,2		44,3		43		44,4
19	відправник	49,8	42,6	50,6	43,2	49,8	42,5	51,6	44
	адресат		42,5		43		42,4		43,9
13	відправник	49	41,9	49,9	42,7	49,2	41,9	50,7	43,5
	адресат		41,7		42,6		41,6		43,4

#### 4.6.2. Топологія де Бруйна

Топологія на основі перетворень де Бруйна і надлишкового коду є добре збалансованою і зв'язною топологією. За рахунок комбінації графів де Бруйна в різних системах числення топологія дозволяє варіювати ступенем вершин, діаметром та кількістю альтернативних шляхів для маршрутизації [88].

Для експерименту використані графи, утворені при об'єднанні дерев з графом на основі кодових перетворень де Бруйна. У топології 01Т створений граф на 9 вершин з потрійним зміщенням кодів в кожную сторону. Для топології 012TZ створений граф на 30 вершин зі зміщенням кодів по п'ять разів в кожную сторону. Якщо зміщення дає унікальний код, що ще не присутній на графі, тоді утворюється нова вершина та зв'язок до неї [89].

Надлишкова топологія де Бруйна 01Т зображена на рис. 4.11 [90].



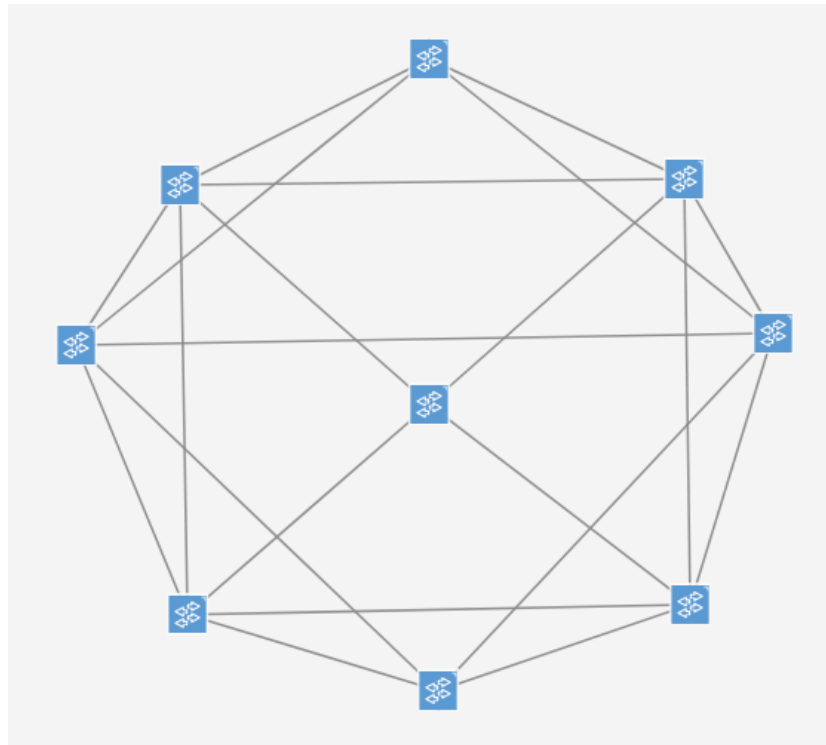


Рис. 4.11. Топологія де Бруйна 01Т у візуалізації контролера *ONOS*

Результати моделювання передачі трафіку для цієї топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.3.

Таблиця 4.3

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для топології де Бруйна 01Т

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
9	відправник	50,8	43,3	51,3	44,1	50,2	42,7	52	44,6
	адресат		43,3		44		42,6		44,5
7	відправник	47	40,1	44,7	38,2	48,3	41,2	49,4	42,3
	адресат		39,9		38		41,1		42,1
5	відправник	41,5	35,4	41,7	35,5	43	37	47,9	40,6
	адресат		35,3		35,4		36,8		40,5

Надлишкова топологія де Бруйна 012TZ зображена на рис. 4.12, результати моделювання передачі трафіку для топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.4.

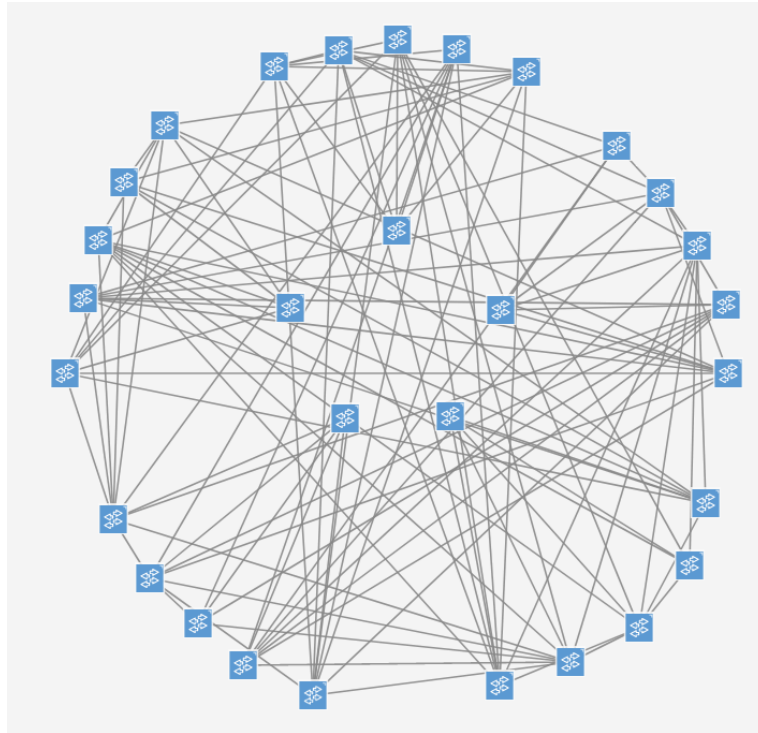


Рис. 4.12. Топологія де Бруйна 012TZ у візуалізації контролера *ONOS*

Таблиця 4.4

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для топології де Бруйна 012TZ

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
30	відправник	45	38,7	44,3	37,9	42,5	36,6	45,4	38,9
	адресат		38,6		37,7		36,5		38,8
23	відправник	40,4	34,5	38,8	33,3	39,1	33,5	42,7	36,7
	адресат		34,4		33,2		33,2		36,5
16	відправник	35,2	30,1	33,6	28,7	34,3	29,3	40,5	34,6
	адресат		30		28,6		29,2		34,5

### 4.6.3. Топологія тор

З геометричної точки зору тор створюється обертанням кола навколо осі, що лежить у площині цього кола і не перетинає її. У комп'ютерних мережах цю топологію представляють як сітчасте з'єднання з вузлами, розташованими у прямокутному масиві розмірністю 2 та більше. Кожен вузол під'єднаний до найближчих сусідів і відповідних вузлів на протилежних краях масиву. Відповідно, у цій решітці кожен вузол має  $2N$  з'єднань. Топологія тор названа на честь сформованої таким чином решітки, яка топологічно однорідна  $N$ -вимірному тору.

Для експерименту була використана вбудована у *mininet* утиліта *--topo*, яка дозволяє, зокрема, побудувати віртуальну мережу з топологією тор зазначеної розмірності. Були обрані варіанти торів 3x3, 6x6 та 9x9 із, відповідно, кількістю вершин 9, 36 та 81. Додання більшої кількості вершин та зв'язків унеможливлювалось потужностями апаратного забезпечення, що використовувалось для експерименту [90].

Топологія тор 3x3 зображена на рис. 4.13, результати моделювання передачі трафіку для топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.5.

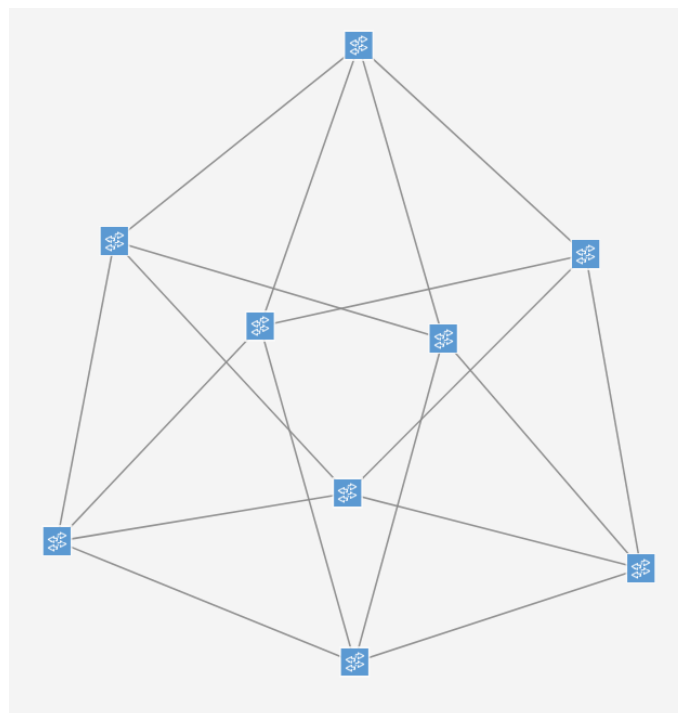


Рис. 4.13. Топологія тор 3x3 у візуалізації контролера *ONOS*

Таблиця 4.5

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для топології  
тор 3х3

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
9	відправник	51	44	49,5	42,3	48,8	41,6	51,7	44,8
	адресат		43,7		42,2		41,4		44,6
7	відправник	45,2	38,4	42	36,1	43,5	37,3	47,8	40,3
	адресат		38,3		36		37,2		40,2
5	відправник	33,9	29,1	33,6	28,5	35,1	30,2	42,6	36,2
	адресат		28,9		28,3		30		36,1

Топологія тор 6х6 зображена на рис. 4.14, результати моделювання передачі трафіку для топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.6.

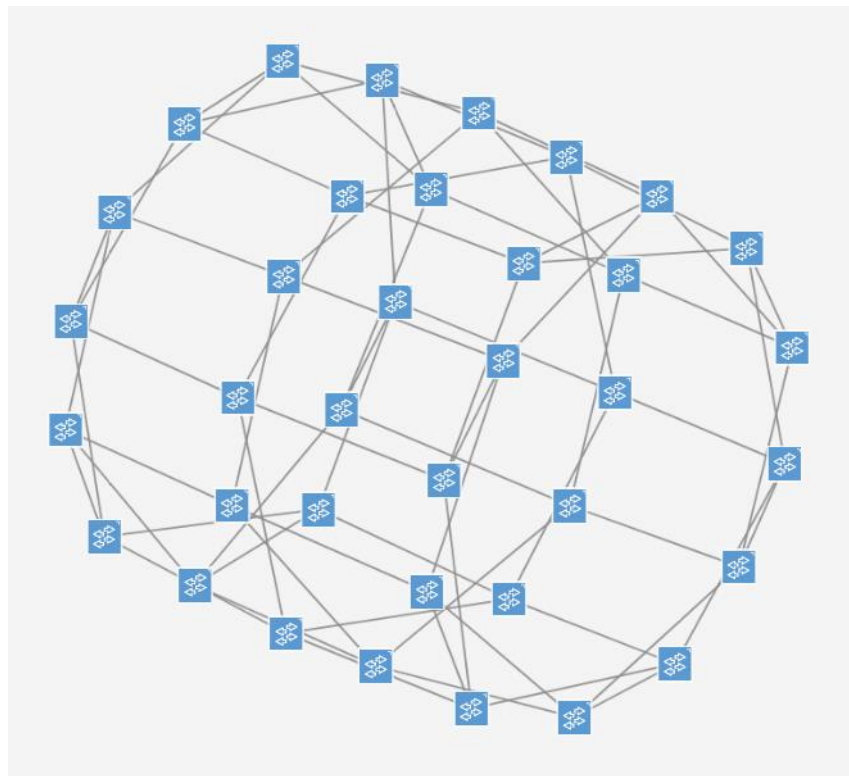


Рис. 4.14. Топологія тор 6х6 у візуалізації контролера *ONOS*

Таблиця 4.6

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для топології тор 6х6

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
36	відправник	44,1	37,6	42,6	36,6	43,7	37,4	45,2	38,5
	адресат		37,5		36,4		37,3		38,4
27	відправник	35,5	30,4	32,4	27,6	35,6	30,5	41,2	35,4
	адресат		30,3		27,5		30,2		35,3
18	відправник	24,5	21,3	22,9	19,5	26	22,2	33,6	28,8
	адресат		21,1		19,3		22		28,6

Топологія тор 9х9 зображена на рис. 4.15, результати моделювання передачі трафіку для топології з поступовим відключенням комутаторів продемонстровані у таблиці 4.7.

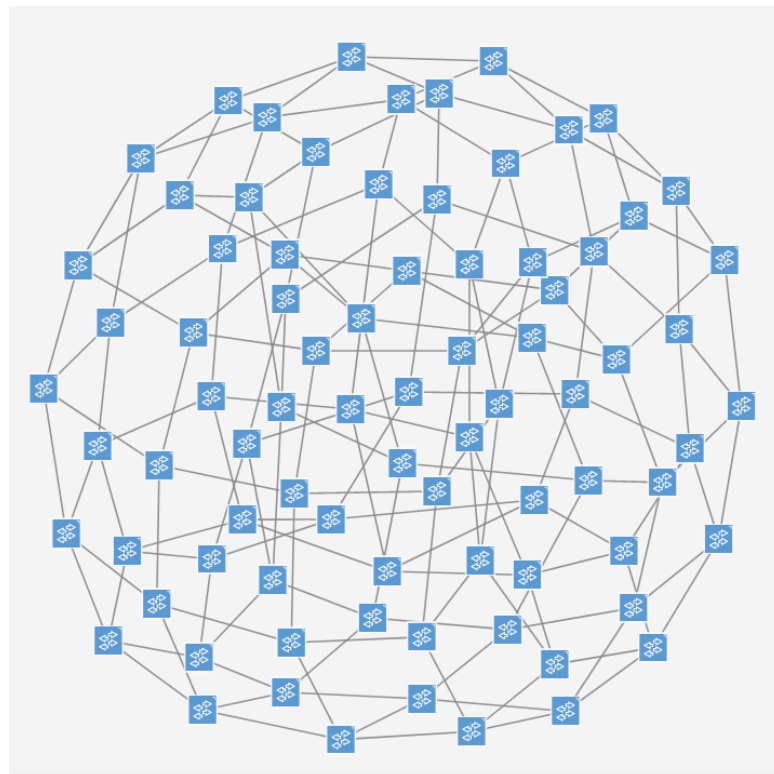


Рис. 4.15. Топологія тор 9х9 у візуалізації контролера *ONOS*

Таблиця 4.7

Порівняльна статистика об'єму (О) та швидкості (Ш) трафіку для топології  
тор 9х9

Вузли	Тип хосту	<i>Reactive Forwarding</i>		<i>Host Intent Forwarding</i>		<i>Point Intent Forwarding</i>		<i>Методи, що пропонуються</i>	
		О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)	О (ГБ)	Ш (Гбіт/с)
81	відправник	38,4	33	37,6	31,9	37,9	32,4	40,1	34,1
	адресат		32,8		31,9		32,3		34
61	відправник	26,7	23,7	24,9	21,3	29,6	25,5	37,4	31,7
	адресат		23,6		21,1		25,3		31,5
41	відправник	15,6	13,5	14,3	12,3	16	13,7	25,2	20,3
	адресат		13,3		12,2		13,6		20,2

#### 4.7. Аналіз отриманих результатів моделювання

Порівняльні графіки для віртуальної мережі повнозв'язної топології зображені на рис. 4.16-4.17.

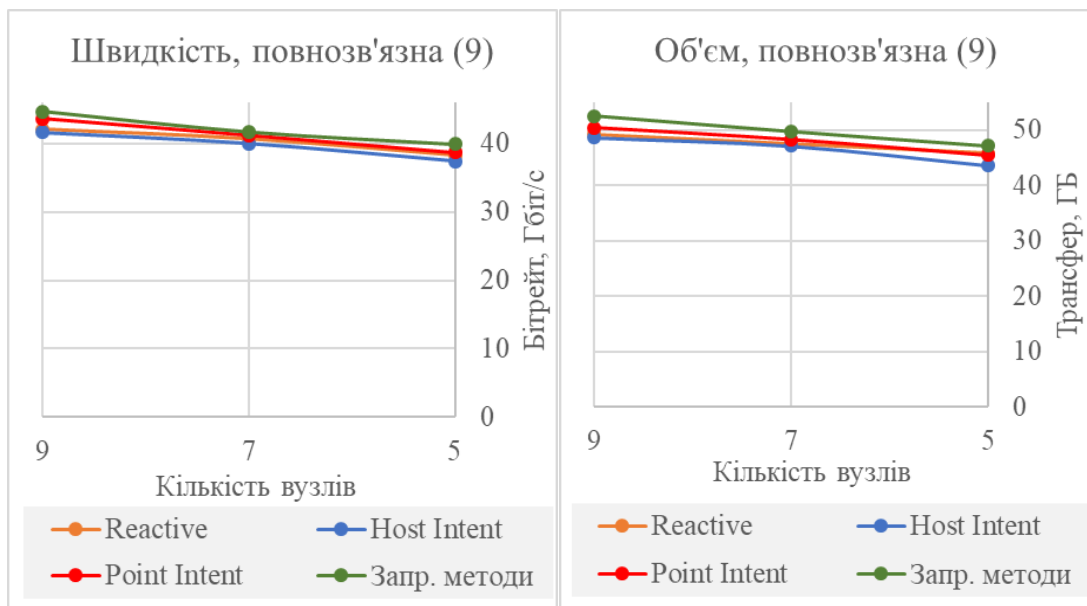


Рис. 4.16. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для повнозв'язної топології на 9 вершин

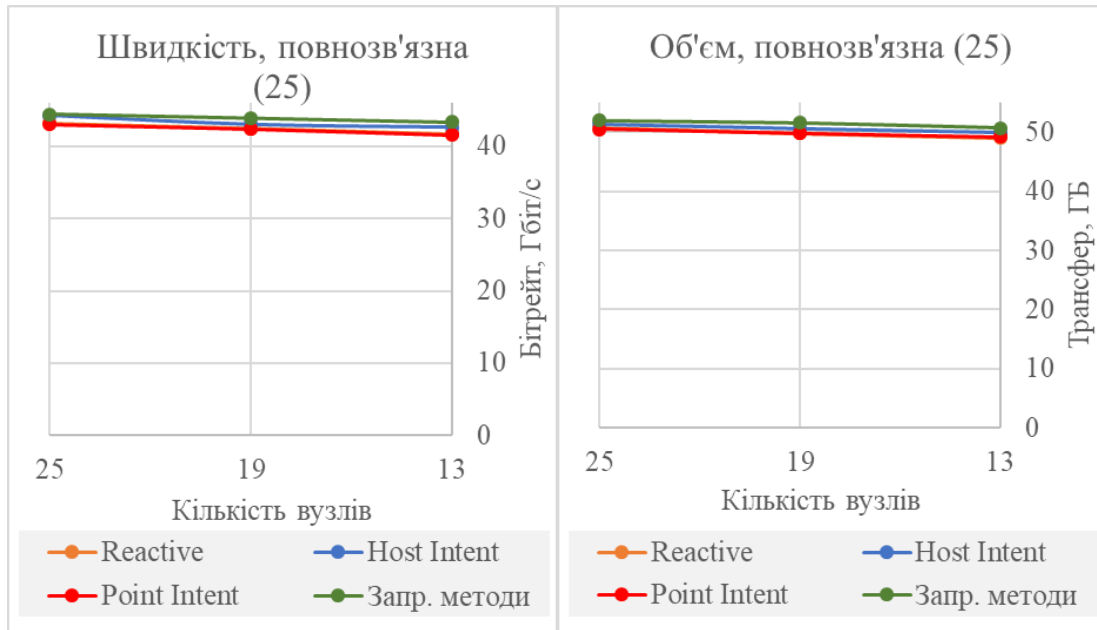


Рис. 4.17. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для повнозв'язної топології

Згідно результатів моделювання були побудовані порівняльні графіки падіння швидкості передачі даних та їхнього об'єму для кожної топології та в залежності від методу, за допомогою якого відбувалась побудова каналів зв'язку.

З отриманих даних слідує, що пропускна здатність мережі змінилася незначно: для графу з 9 вершинами існуючі методи побудови віртуальних каналів зв'язку продемонстрували падіння швидкості передачі даних близько 10%. У той же час метод, що пропонується, дав падіння швидкості 6,5%, що краще за методи, з якими він порівнюється. Для графу з 25 вершинами падіння швидкості та об'єму виявилось ще більш незначним: 4,5% для існуючих методів та 3,5% для того, що пропонується. В цілому, можна зробити висновок, що для топології мережі з максимальною зв'язністю запропоновані методи продемонстрували свої можливості стримано.

На графі, побудованому на основі перетворень Де Бруйна, що має доволі високу зв'язність, метод проявив себе краще. Падіння швидкості для трьох існуючих методів побудови каналів зв'язку становить в середньому 18,5%. В той

же час методи, що пропонується, продемонстрували падіння об'єму на 7,8% та швидкості на 9,2%.

Порівняльні графіки для віртуальної мережі топології де Бруйна 01T та 012TZ зображені на рис. 4.18-4.19.

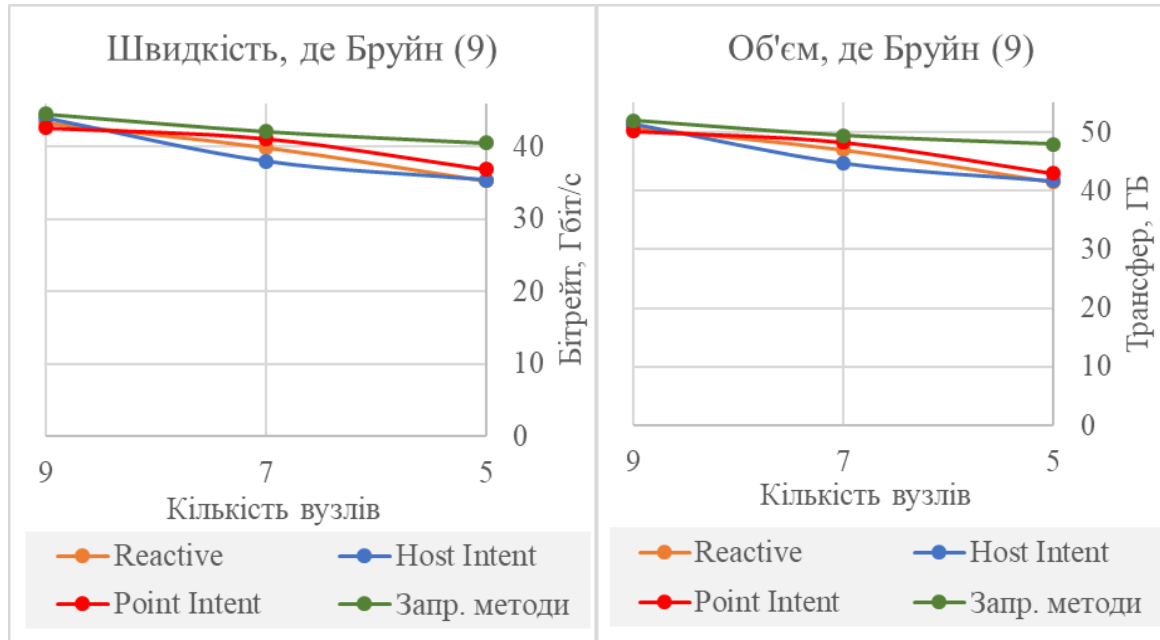


Рис. 4.18. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для топології де Бруйна 01T

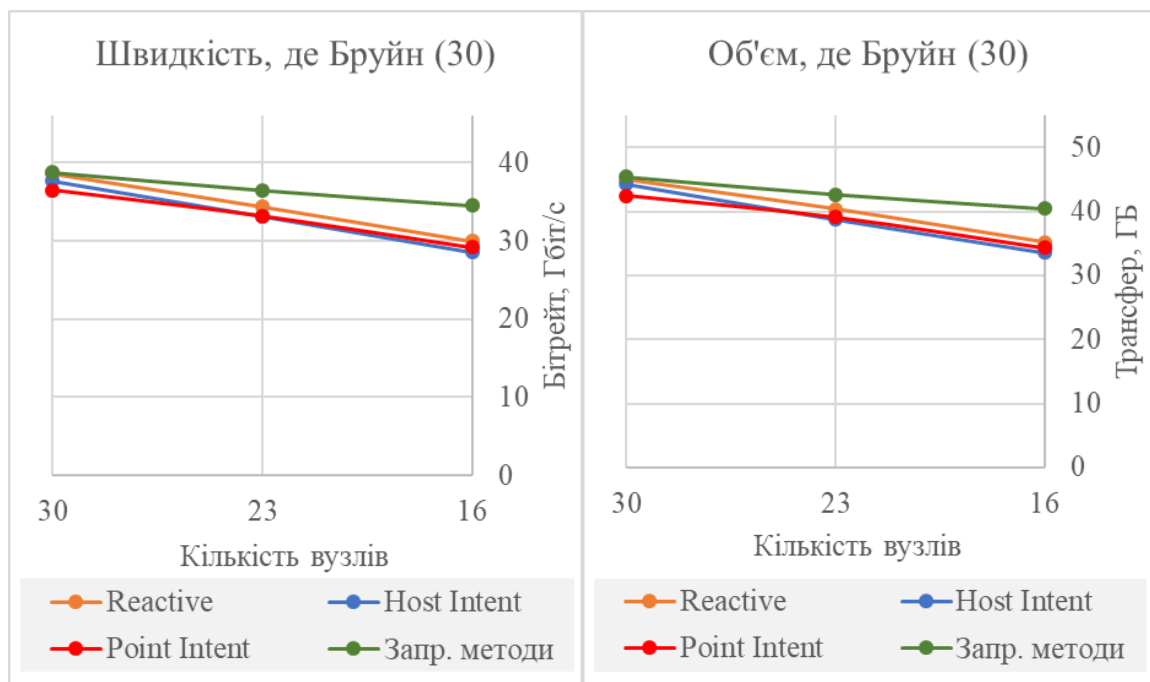


Рис. 4.19. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для топології де Бруйна 012TZ



Ще краще переваги методів побудови каналів зв'язку та динамічної реконфігурації, запропонованих у роботі, стали очевидні при збільшенні кількості вузлів у мережі. Існуючі методи продемонстрували падіння швидкості та об'єму на 20-25% при відключенні половини вузлів у мережі. В той же час методи, що пропонується, дали падіння цих величин лише на 11,3% та 10,8% відповідно.

Подібні результати свідчать про те, що при застосуванні запропонованих методів швидкість передачі трафіку підтримується практично на одному рівні навіть коли переваги зв'язності топології поступово втрачаються. Це також свідчить про те, що контролер *SDN* своєчасно переналаштовує мережу після збою кожного вузла, що дозволяє демонструвати стабільність передачі даних.

Найкраще запропоновані методи проявили свої переваги при перевірці на віртуальній мережі найменш зв'язної топології з тих, що перевірялася – тор. Спостерігалась значна різниця у падіннях швидкості між існуючими методами побудови каналів зв'язку у контролері *ONOS* та тими, що пропонуються (на основі урахування чотирьох метрик якості обслуговування). При побудові мережі на топології тор 3x3 для існуючих методів падіння швидкості і об'єму склало від 28 до 35%, а для методів, що пропонуються – 19,4 та 17,6% відповідно (рис. 4.20).

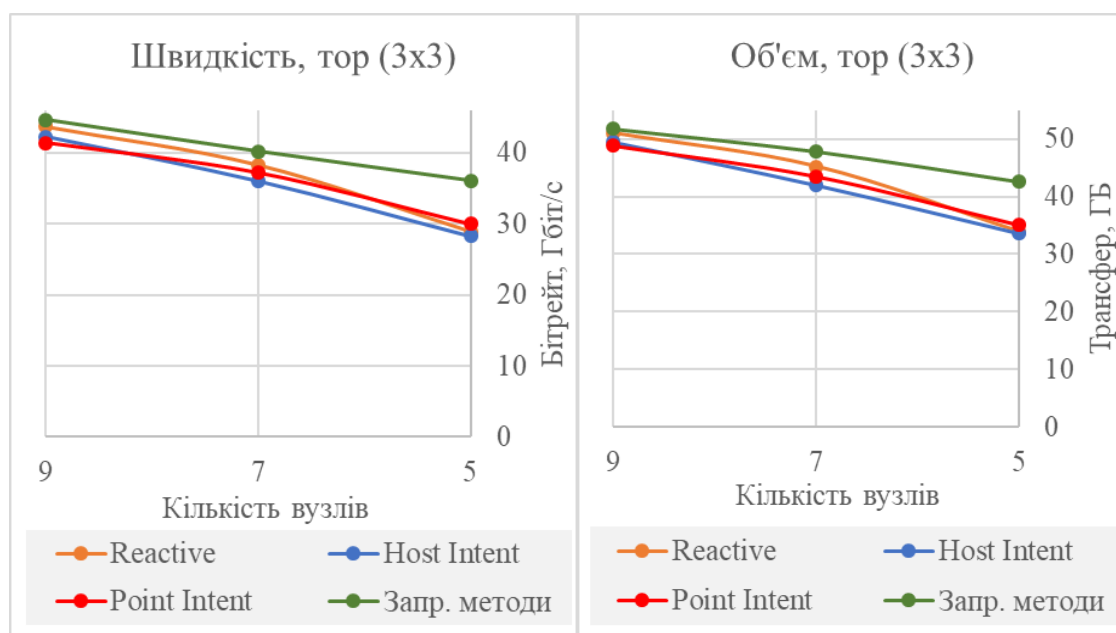


Рис. 4.20. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для топології тор 3x3

Для топології тор 6х6 падіння величин склало відповідно від 40 до 44% для існуючих методів і 25,7% для тих, що пропонуються у даній роботі (рис. 4.21).

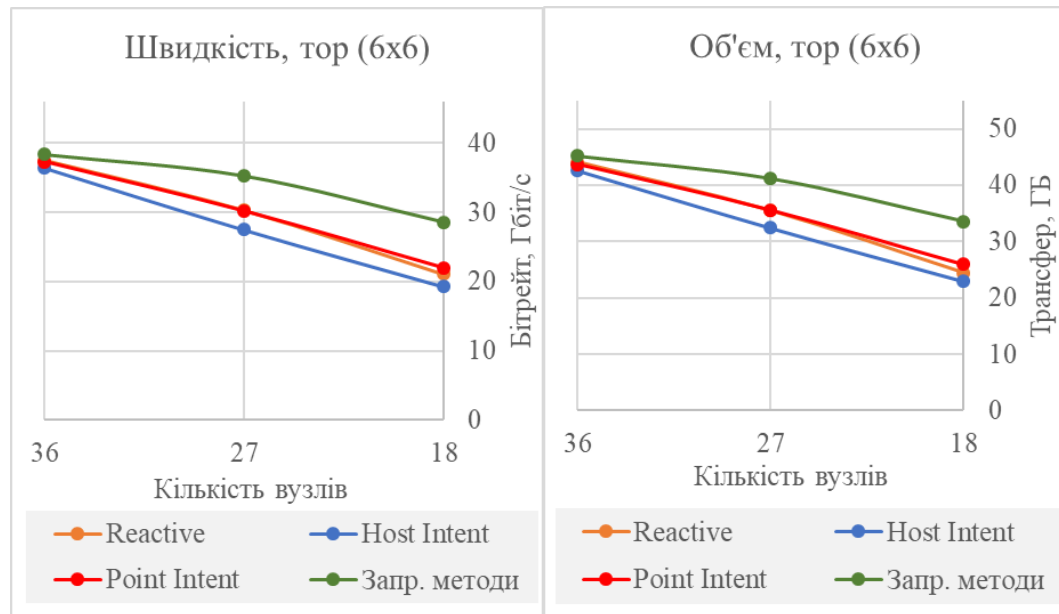


Рис. 4.21. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для топології тор 6х6

Тенденція зберігається і для мережі топології тор 9х9, порівняльні графіки для якої зображені на рис. 4.22.

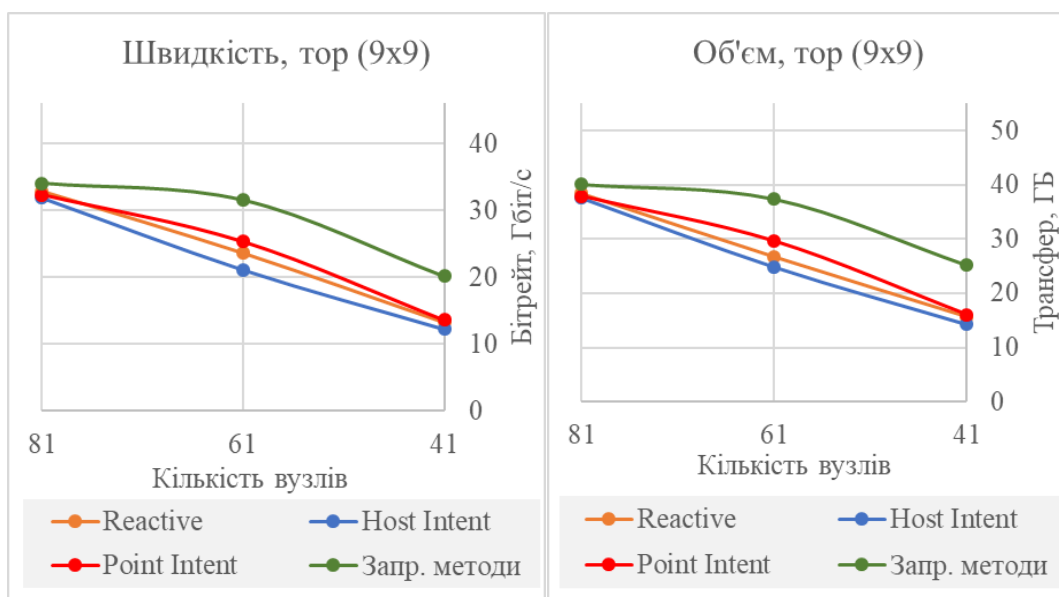


Рис. 4.22. Порівняльні графіки швидкості та об'єму даних при зменшенні кількості вершин для топології тор 9х9

Втрати швидкості передачі даних для існуючих методів для віртуальної мережі цієї топології склали від 58 до 62%, а для методів, що пропонуються – 40,7% (швидкість) та 37,1% (об'єм).

Для топологій обмеженої зв'язності метод з використанням інтегрального критерію продемонстрував перевагу у стабільності передачі даних у віртуальній мережі. Стає очевидним, що при збільшенні загальної кількості вузлів на графі та, відповідно, кількості відключених вузлів, метод продовжує демонструвати суттєву перевагу над існуючими. Це означає, що він буде працювати ефективно при масштабуванні мережі.

Загальна порівняльна статистика об'єму та швидкості переданих даних при моделюванні з використанням існуючих методів побудови каналів зв'язку та того, що пропонується, представлена у табл. 4.8 та 4.9 відповідно.

Таблиця 4.8

## Порівняльна статистика переданого об'єму даних (у ГБ)

Топологія (кількість вузлів)	Вузли	Існуючі методи	Методи, що пропонуються	Різниця
Повнозв'язна (9)	9	49,4	52,5	6,28%
	7	47,6	49,7	4,34%
	5	45,0	47,1	4,74%
Повнозв'язна (25)	25	50,9	52,0	2,16%
	19	50,1	51,6	3,06%
	13	49,4	50,7	2,70%
Де Бруйна (9)	9	50,8	52,0	2,43%
	7	46,7	49,4	5,86%
	5	42,1	47,9	13,87%
Де Бруйна (30)	30	43,9	45,4	3,34%
	23	39,4	42,7	8,28%
	16	34,4	40,5	17,85%

Продовження таблиці 4.8

Тор (3x3)	9	49,8	51,7	3,88%
	7	43,6	47,8	9,72%
	5	34,2	42,6	24,56%
Тор (6x6)	36	43,5	45,2	3,99%
	27	34,5	41,2	19,42%
	18	24,5	33,6	37,33%
Тор (9x9)	81	38,0	40,1	5,62%
	61	27,1	37,4	38,18%
	41	15,3	25,2	64,71%

Таблиця 4.9

Порівняльна статистика швидкості передачі даних (у ГБіт/с)

Топологія (кількість вузлів)	Вузли	Існуючі методи	Методи, що пропонуються	Різниця
Повнозв'язна (9)	9	42,4	44,7	5,34%
	7	40,6	41,7	2,63%
	5	38,1	39,9	4,63%
Повнозв'язна (25)	25	43,5	44,4	2,07%
	19	42,6	43,9	2,97%
	13	42,0	43,4	3,42%
Де Бруйна (9)	9	43,3	44,5	2,77%
	7	39,7	42,1	6,13%
	5	35,8	40,5	13,02%
Де Бруйна (30)	30	37,6	38,8	3,19%
	23	33,6	36,5	8,63%
	16	29,3	34,5	17,88%
Тор (3x3)	9	42,4	44,6	5,11%
	7	37,2	40,2	8,16%

Продовження таблиці 4.9

Top (3x3)	5	29,1	36,1	24,20%
Top (6x6)	36	37,1	38,4	3,60%
	27	29,3	35,3	20,34%
	18	20,8	28,6	37,50%
Top (9x9)	81	32,3	34,0	5,15%
	61	23,3	31,5	35,00%
	41	13,0	20,2	54,99%

У ході експерименту методи, що пропонуються в роботі, були перевірені на кількох топологіях – повнозв'язній, топології, заснованій на перетвореннях Де Бруїна, і торі різних розмірностей. Найкраще методи виявили себе на топології тор, продемонструвавши найбільш стабільне збереження швидкості та кількості переданих даних при поступовому відключенні вузлів. У той же час, повнозв'язна топологія виявила переваги методів найслабше. Це дозволяє припустити, що при побудові комп'ютерної мережі на основі топології великої зв'язності відмовостійкість буде забезпечена лише геометричними особливостями графу мережі. Однак при побудові мережі на основі топологій відносно невеликої зв'язності, відмовостійкість каналів зв'язку буде додатково забезпечена запропонованими у роботі методами. Оскільки побудова великомасштабної повнозв'язної мережі на практиці зазвичай технічно неможлива або дорога, запропоновані методи дозволять досягти кращих результатів щодо передачі трафіку при організації відносно слабозв'язаної мережі.

## Висновки до розділу 4

Проведено огляд існуючих програмних реалізацій *SDN*-контролерів. Зроблено вибір контролеру *ONOS* для виконання моделювання віртуальної мережі *SDN* та перевірки запропонованих методів. Вибір обґрунтований тим, що *ONOS* має достатню для виконання поставленої задачі функціональність та висуває прийнятні для проведення експерименту апаратні вимоги. На базі *ONOS* створено застосунок, що дозволяє побудувати віртуальну мережу, яка використовує побудову шляхів передачі трафіку та динамічну реконфігурацію на основі запропонованих у роботі методів.

Моделювання віртуальної мережі *SDN* під управлінням контролера *ONOS* з використанням запропонованих у розділах 2 і 3 методів побудови каналів зв'язку та динамічної реконфігурації продемонструвало безумовну перевагу перед тими методами побудови каналів зв'язку, що існували раніше.

В ході моделювання були проаналізовані результати передачі трафіку для декількох топологій мережі різної зв'язності. Для топології з повною зв'язністю запропоновані методи продемонстрували невелику різницю у падінні швидкості передачі даних при поступовому виході з ладу вузлів мережі, а об'єм переданих даних був більшим за існуючі методи всього на 2-5%. Це означає, що переваги повнозв'язності переважають недоліки існуючих методів побудови шляхів, що дозволяє їм демонструвати прийнятні результати швидкості передачі трафіку.

Рішення на основі запропонованих методів проявило себе краще для менш зв'язних топологій. Для топології на основі надлишкового коду де Бруїна падіння швидкості для існуючих методів побудови каналів зв'язку становило в середньому 18,5%, а для методу, що пропонується – на 9,2%; об'єм переданих даних був до 18% більшим. Для топології тор падіння швидкості для існуючих методів побудови каналів зв'язку становило до 60%, а для методу, що пропонується – не перевищило 41%; об'єм переданих даних був до 64,7% більшим. Це свідчить про те, що урахування параметрів якості обслуговування дозволяє підтримувати прийнятний рівень швидкості передачі даних навіть при відключенні значної кількості вузлів,

що неможливе для існуючих методів побудови каналів зв'язку, які враховують лише одну або дві метрики.

Експеримент підтвердив правильність обраних технічних засобів розробки застосунку для *SDN*-контролера *ONOS* та перевагу запропонованих методів. Висока швидкість реконфігурації мережі, що дозволила підтримувати стабільний рівень передачі даних, свідчить про те, що модифікований застосунок не перевантажує апаратні ресурси контролера.

## ВИСНОВКИ

У дисертаційній роботі виконано теоретичне обґрунтування та одержано нове рішення задачі підвищення швидкості та надійності передачі даних в комп'ютерних мережах. З цією метою розроблено новий комплексний спосіб побудови віртуальної комп'ютерної мережі на основі технології *SDN* для передачі заданого типу трафіку. Розроблений спосіб базується на інтеграції технологій віртуальних мереж та *SDN* та складається із методу побудови каналів зв'язку на основі інтегрального критерію, методу динамічної реконфігурації з використанням резервних каналів зв'язку, та програмних засобів реалізації цих методів на контролері *SDN*.

Основні наукові та практичні результати дисертаційного дослідження полягають у наступному:

1. Досліджено існуючі рішення побудови віртуальних комп'ютерних мереж з використанням технології *SDN*. На основі аналізу літературних джерел зроблено постановку задачі про необхідність розробки комплексного рішення побудови віртуальної комп'ютерної мережі на основі технології *SDN*, яке дозволить вдосконалити процеси передачі даних у мережі у випадку відмов окремих її ділянок та забезпечити відповідність мережі якості обслуговування. З цією метою був запропонований спосіб побудови віртуальної мережі на основі технології *SDN*, що складається із запропонованих у роботі методу побудови каналів зв'язку на основі інтегрального критерію з урахуванням метрик якості обслуговування і критерію надійності комутаторів, методу динамічної реконфігурації з використанням раніше побудованих каналів зв'язку, та програмних засобів реалізації цих методів на контролері *SDN*.

2. Розроблено та обґрунтовано інтегральний критерій побудови каналів зв'язку у віртуальній комп'ютерній мережі на основі технології *SDN*, який, на відміну від існуючих способів побудови, одночасно враховує необхідну кількість та пріоритети метрик якості обслуговування каналів зв'язку мережі, що дозволяє досягти збереження стабільної швидкості передачі трафіку будь-якого типу.



3. Розроблено та обґрунтовано критерій надійності вузлів мережі для побудови каналів зв'язку, який, на відміну від існуючих способів побудови, враховує здатність проміжних пристроїв мережі транспортувати трафік між своїми портами, що дозволяє збільшити надійність каналів зв'язку та зменшити втрати даних під час передачі.

4. Розроблено та обґрунтовано метод побудови відмовостійких каналів зв'язку віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на розділенні віртуальної мережі на підмережі та використовує розроблені інтегральний критерій побудови каналів зв'язку з урахуванням якості обслуговування та критерій надійності вузлів, що дозволяє забезпечити стабільну швидкість, зменшити затримку та втрати даних під час передачі будь-якого типу трафіку у разі виходу з ладу окремих ділянок мережі. За рахунок використання методу досягнуто збереження швидкості передачі даних та не допущено її зниження більше ніж на 41% після відключення половини вузлів.

5. Удосконалено метод динамічної реконфігурації віртуальної мережі на основі технології *SDN*, який, на відміну від існуючих методів, базується на використанні відмовостійких резервних каналів зв'язку, збережених у пам'яті контролера *SDN*, що дозволяє підтримувати безперервний сеанс передачі трафіку у разі відмови основного каналу зв'язку. Завчасна побудова та вибір резервних каналів зв'язку, що не перетинаються, дозволили підтримувати безперебійний сеанс передачі трафіку та збільшити об'єм переданих даних на 64,7% при відключенні половини вузлів мережі.

6. Розроблено програмне забезпечення для *SDN* контролера *ONOS*, що реалізує запропонований спосіб побудови віртуальної *SDN* мережі та базується на використанні запропонованих у роботі методів. Експериментальне моделювання довело ефективність способу на різних топологічних організаціях, швидкість передачі після відключення половини комутаторів збільшилась, порівняно з існуючими методами, на 5% для повнозв'язної організації, та на 40-60% – для організацій більш низької зв'язності (Де Бруйна, *top*).

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. J. Howarth, "Internet Traffic from Mobile Devices (Jan 2024)," *Exploding Topics*. Accessed: Jan. 28, 2024. [Online]. Available: <https://explodingtopics.com/blog/mobile-internet-traffic>
2. Q. Waseem, W. I. S. Wan Din, A. Aminuddin, M. H. Mohammed, and R. F. A. Aziza, "Software-Defined Networking (SDN): A Review," in *2022 5th Int. Conf. on Information and Communications Technology (ICOIACT)*, IEEE, Aug. 2022, pp. 30–35. doi: 10.1109/ICOIACT55506.2022.9972067.
3. M. Sanchez, "Why distributed cloud networking is the future of enterprise networks," *SDX Central*. Accessed: Oct. 10, 2024. [Online]. Available: <https://www.sdxcentral.com/articles/contributed/why-distributed-cloud-networking-is-the-future-of-enterprise-networks/2023/11/>
4. S. K. Mohindroo, "Unraveling the Power of Software-Defined Networking (SDN) Past, Present, and Future," *Staying Alive*. Accessed: Oct. 10, 2024. [Online]. Available: <https://blog.stayingalive.in/strategic-innovations/unraveling-the-power-of.html>
5. N. Wang, Z.-Y. Jin, and J. Zhao, "Cascading failures of overload behaviors on interdependent networks," *Physica A: Statistical Mechanics and its Applications*, vol. 574, p. 125989, Jul. 2021. doi: 10.1016/j.physa.2021.125989.
6. H. M. Do, M. A. Gregory, and S. Li, "SDN-based wireless mobile backhaul architecture: Review and challenges," *J. Network and Computer Applications*, vol. 189, p. 103138, Sep. 2021. doi: 10.1016/j.jnca.2021.103138.
7. J. Fruhlinger, "What is a virtual network," *NetworkWorld*. Accessed: Feb. 08, 2024. [Online]. Available: <https://www.networkworld.com/article/971891/what-is-a-virtual-network.html>
8. L. Wang, Z. Lu, X. Wen, G. Cao, X. Xia, and L. Ma, "An SDN-based seamless convergence approach of WLAN and LTE networks," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*,

- Chongqing, China: IEEE, May 2016, pp. 944–947. doi: 10.1109/ITNEC.2016.7560501.
9. Z. Shu et al., "Construction of SDN Network Management Model Based on Virtual Technology Application," 2022, pp. 257–268. doi: 10.1007/978-981-19-2456-9\_28.
  10. F. H. P. Fitzek, F. Granelli, and P. Seeling, *Computing in Communication Networks*, Elsevier, 2020. doi: 10.1016/C2019-0-01898-5.
  11. G. Yang, H. Jin, M. Kang, G. J. Moon, and C. Yoo, "Network Monitoring for SDN Virtual Networks," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, IEEE, Jul. 2020, pp. 1261–1270. doi: 10.1109/INFOCOM41043.2020.9155260.
  12. D. Barton, "ONF Announces New Sustainable Mobile and RAN Transformation (SMaRT) 5G Project," *Open Networking Foundation*. Accessed: Oct. 10, 2024. [Online]. Available: <https://opennetworking.org/news-and-events/blog/onf-announces-new-sustainable-mobile-and-ran-transformation-smart-5g-project/>
  13. K. Surya, "Architecture of Software Defined Networks (SDN)," *GeekForGeeks*. Accessed: Oct. 10, 2024. [Online]. Available: <https://www.geeksforgeeks.org/architecture-of-software-defined-networks-sdn/>
  14. K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): A survey," *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, Dec. 2017. doi: 10.1002/sec.1737.
  15. F. Bannour, S. Souihi, and A. Mellouk, "Adaptive distributed SDN controllers: Application to Content-Centric Delivery Networks," *Future Generation Computer Systems*, vol. 113, pp. 78–93, Dec. 2020. doi: 10.1016/j.future.2020.05.032.
  16. A. Nunez, J. Ayoka, Z. Islam, and P. Ruiz, "Software Defined Networking: An Overview," Feb. 2023. doi: <https://doi.org/10.48550/arXiv.2302.00165>.
  17. N. Khan, R. bin Salleh, A. Koubaa, Z. Khan, M. K. Khan, and I. Ali, "Data plane failure and its recovery techniques in SDN: A systematic literature review," *J. King Saud Univ. - Computer and Information Sciences*, vol. 35, no. 3, pp. 176–201, Mar. 2023. doi: 10.1016/j.jksuci.2023.02.001.

18. A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: A survey," *J. Supercomput.*, vol. 76, no. 10, pp. 7545–7593, Oct. 2020. doi: 10.1007/s11227-020-03180-7.
19. M. R. Adrian, D. H. Kurniawan, A. Faza, J. Maulina, and M. R. Shihab, "A Brief Look at Software Defined Network (SDN) Implementation: Gaining Benefits and Coping with the Challenges at a Telecommunication Company," *IOP Conference Series: Materials Science and Engineering*, vol. 879, no. 1, p. 012046, Jul. 2020, doi: 10.1088/1757-899X/879/1/012046.
20. A. de Talhouët, "The evolution of Software Defined Networking," *Red Hat*. [Online]. Available: <https://www.redhat.com/en/blog/evolution-software-defined-networking>. Accessed: Oct. 10, 2024.
21. D. Varnum, "OpenFlow - Basic Concepts and Theory," *Overlaid*. [Online]. Available: <https://overlaid.net/2017/02/15/openflow-basic-concepts-and-theory/>. Accessed: Jan. 31, 2024.
22. S. K. Keshari, V. Kansal, and S. Kumar, "A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)," *Wireless Personal Communications*, vol. 116, no. 3, pp. 2593–2614, Feb. 2021, doi: 10.1007/s11277-020-07812-2.
23. A. Schwabe, E. Rojas, and H. Karl, "Minimizing downtimes: Using dynamic reconfiguration and state management in SDN," in *Proc. IEEE Conf. Network Softwarization (NetSoft)*, Jul. 2017, pp. 1–5, doi: 10.1109/NETSOFT.2017.8004209.
24. A. Bianco, P. Giaccone, R. Mashayekhi, M. Ullio, and V. Vercellone, "Scalability of ONOS reactive forwarding applications in ISP networks," *Computer Communications*, vol. 102, pp. 130–138, Apr. 2017, doi: 10.1016/j.comcom.2016.09.007.
25. Y. Han et al., "ONVisor: Towards a scalable and flexible SDN-based network virtualization platform on ONOS," *International Journal of Network Management*, vol. 28, no. 2, Mar. 2018, doi: 10.1002/nem.2012.

26. S. Abdallah, A. Kayssi, I. H. Elhajj, and A. Chehab, "Network Convergence in SDN Versus OSPF Networks," in *Proc. 5th Int. Conf. Software Defined Systems (SDS)*, Barcelona, Spain, Apr. 2018, pp. 130–137, doi: 10.1109/SDS.2018.8370434.
27. D. Sanvito, D. Moro, M. Gulli, I. Filippini, A. Capone, and A. Campanella, "Enabling external routing logic in ONOS with Intent Monitor and Reroute service," in *Proc. 4th IEEE Conf. Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 332–334, doi: 10.1109/NETSOFT.2018.8460042.
28. D. Sanvito, D. Moro, M. Gulli, I. Filippini, A. Capone, and A. Campanella, "ONOS Intent Monitor and Reroute service: enabling plug-and-play routing logic," in *Proc. 4th IEEE Conf. Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 272–276, doi: 10.1109/NETSOFT.2018.8460064.
29. T. Irfan, R. Hakimi, A. C. Risdianto, and E. Mulyana, "ONOS Intent Path Forwarding using Dijkstra Algorithm," in *Proc. Int. Conf. Electrical Engineering and Informatics (ICEEI)*, Jul. 2019, pp. 549–554, doi: 10.1109/ICEEI47359.2019.8988853.
30. A. Rafiq, A. Mehmood, and W.-C. Song, "Intent-Based Slicing between Containers in SDN Overlay Network," *Journal of Communications*, vol. 15, no. 3, pp. 237–244, Mar. 2020, doi: 10.12720/jcm.15.3.237-244.
31. M. F. Hyder, T. Fatima, and S. Arshad, "Digital forensics framework for intent-based networking over software-defined networks," *Telecommunication Systems*, vol. 85, no. 1, pp. 11–27, Jan. 2024, doi: 10.1007/s11235-023-01064-8.
32. Z. Yang and K. L. Yeung, "Minimum weight controller tree design in SDN," *Computer Networks*, vol. 165, p. 106949, Dec. 2019, doi: 10.1016/j.comnet.2019.106949.
33. E. Osei Kofi and E. Ahene, "Enhanced network load balancing technique for efficient performance in software defined network," *PLoS One*, vol. 18, no. 4, p. e0284176, Apr. 2023, doi: 10.1371/journal.pone.0284176.
34. Y. Kulakov, A. Kohan, Y. Pavlenkova, N. Pastrello, and N. Machekhin, "Traffic engineering in a software-defined network based on the decision-making method," *Eastern-European Journal of Enterprise Technologies*, vol. 2, no. 9 (98), pp. 23–28, Apr. 2019, doi: 10.15587/1729-4061.2019.164686.

35. Y. Kulakov, A. Kohan, and Y. Hrabovenko, "Multipath Routing in Intelligent Transport Networks," in *Proc. International Conference on Transport Networks*, 2022, pp. 81–90, doi: 10.1007/978-3-031-04809-8\_7.
36. M. K. Sepehrifar, A. Fanian, and M. B. Sepehrifar, "Shortest Path Computation in a Network with Multiple Destinations," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 3223–3231, Apr. 2020, doi: 10.1007/s13369-020-04340-w.
37. S. Ray and E. Zurich, "A Constraint Based K-Shortest Path Searching Algorithm for Software Defined Networking," 2020. [Online]. Available: <https://www.researchgate.net/publication/341233705>. Accessed: Oct. 10, 2024.
38. Ю. Кулаков та В. Щур, «Спосіб балансування трафіку в SDN мережах на основі модифікованого протоколу маршрутизації за вектором дистанції», *Проблеми інформатизації та управління*, вип. 2, №. 74, сс. 62–67, Черв. 2023, doi: 10.18372/2073-4751.74.17883.
39. Y. Kulakov, S. Kopychko, and I. Hrabovenko, "Adaptive Routing Method in Scalable Software-defined Mobile Networks," in *Proc. International Conference on Software-Defined Mobile Networks*, 2022, pp. 304–313, doi: 10.1007/978-3-031-04812-8\_26.
40. S. Faezi and A. Shirmarz, "A Comprehensive Survey on Machine Learning using in Software Defined Networks (SDN)," *Human-Centric Intelligent Systems*, vol. 3, no. 3, pp. 312–343, Jun. 2023, doi: 10.1007/s44230-023-00025-3.
41. H. Hu, Q. Kang, J. Wang, S. Zhao, Y. Yuan, and Y. Fu, "SDN routing strategy based on genetic algorithm and particle swarm optimization under SFC background," *SPIE-International Society for Optical Engineering*, Apr. 2023, p. 46, doi: 10.1117/12.2673323.
42. R. Yujie, W. Muqing, and C. Yiming, "An Effective Controller Placement Algorithm Based on Clustering in SDN," in *Proc. 6th IEEE International Conference on Computer and Communications (ICCC)*, Dec. 2020, pp. 2294–2299, doi: 10.1109/ICCC51575.2020.9345045.
43. S. Mohanty, B. Sahoo, and S. S. Behera, "An assessment of nature-inspired metaheuristic algorithms for resilient controller placement in software-defined

- networks,” *Decision Analytics Journal*, vol. 12, p. 100501, Sep. 2024, doi: 10.1016/j.dajour.2024.100501.
44. S. Noda, T. Sato, and E. Oki, “Fault-Tolerant Controller Placement Model Considering Load-Dependent Sojourn Time in Software-Defined Network,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4887–4908, Dec. 2023, doi: 10.1109/TNSM.2023.3284830.
  45. P. E. N, J. Kang, K. M. Yusof, J. Bin Din, and S. Kim, “Towards Scalability of Dense Sensor Networks: A Software-Defined Networking Approach,” in *Proc. 2023 IEEE 16th Malaysia International Conference on Communication (MICC)*, Dec. 2023, pp. 1–5, doi: 10.1109/MICC59384.2023.10419610.
  46. H. Xu, X. Chai, and H. Chen, “A Collaborative Approach based on Competitive Game for Multi-Controller Placement in SDN,” in *Proc. 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2022, pp. 1083–1088, doi: 10.1109/CSCWD54268.2022.9776272.
  47. A. Jaiswal, S. S. Patra, B. B. Dash, L. Barik, T. N. Pandey, and M. R. Mishra, “Performance Assessment of Multi-Controller in Software Defined Network,” in *Proc. 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Nov. 2023, pp. 887–892, doi: 10.1109/ICCCIS60361.2023.10425314.
  48. Y. Zhang *et al.*, “Adaptive Digital Twin Placement and Transfer in Wireless Computing Power Network,” *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10924–10936, Mar. 2024, doi: 10.1109/JIOT.2023.3328380.
  49. M. Rafid, H. Ghafoor, and I. Koo, “An SVM-Based Optimal-Controller Selection and Path Selection Protocol for Heterogeneous Communications in SDVNs,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 5, pp. 3828–3842, May 2024, doi: 10.1109/TITS.2023.3326651.
  50. S. Pathak, A. Mani, and A. Chatterjee, “A Cooperative Hybrid Quantum-Inspired Salp Swarm and Differential Evolution for Solving CEC 2022 Benchmark and Controller Placement Problems in Software Defined Networks,” *IEEE Access*, vol. 12, pp. 90331–90344, 2024, doi: 10.1109/ACCESS.2024.3414930.

51. I. O. Adebayo, M. O. Adigun, and P. Mudali, "Neighbourhood Centrality Based Algorithms for Switch-to-Controller Allocation in SD-WANs," in *Proc. 2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, Aug. 2023, pp. 1–6, doi: 10.1109/icABCD59051.2023.10220485.
52. C.-F. Cheng, J. C.-W. Lin, G. Srivastava, and C.-C. Hsu, "Reaching Consensus with Byzantine Faulty Controllers in Software-Defined Networks," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/6662175.
53. W. Feng, C. Liu, B. Cheng, J. Chen, and Z. Wan, "An End-Host-Importance-Aware Secure Service-Enabled Hybrid SDN Deployment," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 2056–2070, Jun. 2023, doi: 10.1109/TNSM.2022.3208695.
54. A. Naseri, M. Ahmadi, and L. PourKarimi, "Placement of SDN controllers based on network setup cost and latency of control packets," *Computer Communications*, vol. 208, pp. 15–28, Aug. 2023, doi: 10.1016/j.comcom.2023.05.015.
55. P. Kumar, S. Bhushan, D. Halder, and A. M. Baswade, "fybrrLink: Efficient QoS-Aware Routing in SDN Enabled Future Satellite Networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2107–2118, Sep. 2022, doi: 10.1109/TNSM.2021.3129876.
56. S. Liu, T. A. Benson, and M. K. Reiter, "Efficient and Safe Network Updates with Suffix Causal Consistency," in *Proc. Fourteenth EuroSys Conference 2019*, New York, NY, USA, ACM, Mar. 2019, pp. 1–15, doi: 10.1145/3302424.3303965.
57. F. Huang, J. Shi, and Q. Xv, "Control path recovery after controller failure in SDN," in *Proc. 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Apr. 2023, pp. 1900–1904, doi: 10.1109/ICSP58490.2023.10248805.
58. A. Al-Shamri and M. Quwaider, "A Combination of Dynamic Load Balancing and ISMDA Architecture using Sharing Data," in *Proc. 2022 13th International*



- Conference on Information and Communication Systems (ICICS)*, Jun. 2022, pp. 93–99, doi: 10.1109/ICICS55353.2022.9811121.
59. P. S. Tiwana and J. Singh, “Software-Defined Networking’s Adaptive Load Optimisation Technique for Multimedia Applications with Multi-Controller Utilisation,” in *Proc. 2023 9th International Conference on Signal Processing and Communication (ICSC)*, Dec. 2023, pp. 167–172, doi: 10.1109/ICSC60394.2023.10441149.
  60. D. A. Khoa, N. T. Kiem, N. T. Kien, N. N. Tuan, and N. H. Thanh, “Traffic-Adaptive Scheme for SDN Control Plane with Containerized Architecture,” in *Proc. 2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, Jan. 2023, pp. 1–6, doi: 10.1109/IMCOM56909.2023.10035625.
  61. H. G. A. Elrahim, N. N. N. Abd Malik, and K. B. M. Yousif, “Revolutionizing Load Management in SDIoT Networks: A Multicriteria Approach for SDN Controller,” in *Proc. 2023 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, Oct. 2023, pp. 13–19, doi: 10.1109/APWiMob59963. 2023.10365605.
  62. M. Hamdan *et al.*, “DPLBAnt: Improved load balancing technique based on detection and rerouting of elephant flows in software-defined networks,” *Computer Communications*, vol. 180, pp. 315–327, Dec. 2021, doi: 10.1016/j.comcom.2021.10.013.
  63. D. I. George Amalarethinam and P. Mercy, “An Analysis on Quality of Service (QoS) Based Routing in Internet of Things (IoT),” *International Journal of Advanced Science and Technology*, vol. 29, no. 5, pp. 488–496, 2020.
  64. M. A. Aulia, A. A. Sukmandhani, and J. Ohliati, “RIP and OSPF Routing Protocol Analysis on Defined Network Software,” in *Proc. 2022 International Electronics Symposium (IES)*, Aug. 2022, pp. 393–397, doi: 10.1109/IES55876.2022. 9888355.
  65. J. E. Gonzalez-Trejo *et al.*, “A Novel Strategy for Computing Routing Paths for Software-Defined Networks Based on MOCeLL Optimization,” *Applied Sciences (Switzerland)*, vol. 12, no. 22, Nov. 2022, doi: 10.3390/app122211590.

66. M. T. Naing, T. T. Khaing, and A. H. Maw, "Evaluation of TCP and UDP Traffic over Software-Defined Networking," in *2019 International Conference on Advanced Information Technologies (ICAIT)*, IEEE, Nov. 2019, pp. 7–12, doi: 10.1109/AITC.2019.8921086.
67. P. P. Ray, "A Survey on Cognitive Packet Networks: Taxonomy, State-of-the-Art, Recurrent Neural Networks, and QoS Metrics," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5663–5683, Sep. 2022, doi: 10.1016/j.jksuci.2021.05.017.
68. M. A. Khan, "A Comprehensive Study of Dijkstra's Algorithm," *SSRN Electronic Journal*, 2023, doi: 10.2139/ssrn.4559304.
69. M. Zhou and N. Gao, "Research on Optimal Path Based on Dijkstra Algorithms," in *Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)*, Paris, France: Atlantis Press, 2019.
70. O. Cherevatenko and Y. Kulakov, "Method of Increasing Data Transmission Stability in Software-Defined Networks Considering Metrics of Quality of Service," *Information, Computing and Intelligent Systems*, no. 4, pp. 37–47, Oct. 2024, doi: 10.20535/2786-8729.4.2024.303467.
71. P. Banerjee, P. Kumar, B. Kumar, and K. Thakur, "A New Proposed Modified Shortest Path Algorithm Using Dijkstra's," in *Proceedings of the 2nd IEEE International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI 2023)*, IEEE, 2023, doi: 10.1109/ACCAI58221.2023.10199281.
72. A. Volokyta *et al.*, "Traffic Engineering with Specified Quality of Service Parameters in Software-Defined Networks," *International Journal of Computer Network and Information Security*, vol. 16, no. 5, pp. 1–13, Oct. 2024, doi: 10.5815/ijcnis.2024.05.01.
73. D. Korenko, O. Cherevatenko, V. Rusinov, and Y. Kulakov, "Creation of the Method of Multipath Routing Using Known Paths in Software-Defined Networks," *Technology Audit and Production Reserves*, vol. 4, no. 2(66), pp. 19–24, Aug. 2022, doi: 10.15587/2706-5448.2022.262787.

74. D. Bhat, J. Anderson, P. Ruth, M. Zink, and K. Keahey, "Application-Based QoE Support with P4 and OpenFlow," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, Apr. 2019, pp. 817–823, doi: 10.1109/INFOCOMW.2019.8845180.
75. J. U. Waziri, "Software-Defined Networking (SDN) Controller for Dynamic Network Traffic Isolation Through OpenFlow," *IRE Journals*, vol. 6, no. 1, 2022.
76. A. Zhu, "OpenFlow Switch: What Is It and How Does It Work?," *Medium*. Accessed: Oct. 10, 2024. [Online]. Available: <https://medium.com/@AriaZhu/openflow-switch-what-is-it-and-how-does-it-work-7589ea7ea29c>
77. A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from OpenFlow to P4: A Survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–37, Sep. 2023, doi: 10.1145/3556973.
78. A. Malik and R. De Fréin, "Graph Modeling for OpenFlow Switch Monitoring," *IEEE Access*, vol. 11, pp. 84543–84553, 2023, doi: 10.1109/ACCESS.2023.3303847.
79. O. Cherevatenko and Y. Kulakov, "Security of Planes and Local Networks with SDN Architecture," in *The International Conference on Security, Fault Tolerance, Intelligence*, 2020, pp. 197–202.
80. L. Peterson, "OpenFlow: Catalyst that Kickstarted the SDN Transformation," *Open Networking Foundation*. Accessed: Oct. 10, 2024. [Online]. Available: <https://opennetworking.org/news-and-events/blog/openflow-catalyst-that-kickstarted-the-sdn-transformation/>
81. K. K. Karmakar *et al.*, "A Trust-Aware OpenFlow Switching Framework for Software-Defined Networks (SDN)," *Computer Networks*, vol. 237, p. 110109, Dec. 2023, doi: 10.1016/j.comnet.2023.110109.
82. "Floodlight 0.4.0 Documentation." Accessed: Oct. 07, 2024. [Online]. Available: <https://floodlight.readthedocs.io/en/latest/modules/core/core.html>
83. "OpenDaylight Documentation, Potassium Documentation." Accessed: Oct. 07, 2024. [Online]. Available: <https://docs.opendaylight.org/en/stable-potassium/>
84. "ONOS Documentation." Accessed: Oct. 07, 2024. [Online]. Available: <https://wiki.onosproject.org/display/ONOS/ONOS+Documentation>

85. R. Alsheikh, E. Fadel, and N. Akkari, “An Adaptive State Consistency Architecture for Distributed Software-Defined Network Controllers: An Evaluation and Design Consideration,” *Applied Sciences*, vol. 14, no. 6, p. 2627, Mar. 2024, doi: 10.3390/app14062627.
86. Ю. Кулаков та О. Череватенко, «Моделювання маршрутизації з використанням відомих маршрутів за допомогою SDN-контролера ONOS,» *Проблеми інформатизації та управління*, вип. 2, №. 74, сс. 55–61, Черв. 2023, doi: 10.18372/2073-4751.74.17882.
87. O. Cherevatenko and Y. Kulakov, “Analysis of Technologies of Software-Defined Networks,” in *Security, Fault Tolerance, Intelligence: Proceedings of the International Conference ICSFTI2019*, Kyiv, Ukraine, May 2019, pp. 188–195.
88. В. Русінов, Б. Іваніщев, А. Антонюк, А. Волокита, та Г. Луцький, «Способи синтезу топологічних організацій на основі кодових перетворень Дебруйна,» *Технічні науки та технології*, вип. 4, №. 22, сс. 131–143, 2020.
89. О. Гончаренко та О. Череватенко, «Способи мультиканальної маршрутизації в мережах надлишкового Де Бруйна,» *Технічні науки та технології*, вип. 2(24), сс. 123–130, 2021, doi: 10.25140/2411-5363-2021-2(24)-123-130.
90. A. Volokyta *et al.*, “Fault Tolerance Exploration and SDN Implementation for de Bruijn Topology Based on Betweenness Coefficient,” *International Journal of Computer Network and Information Security*, vol. 16, no. 1, pp. 97–112, Feb. 2024.

## ДОДАТОК А

### Частина програмного коду

```

public class DijkstraGraphSearch<V extends Vertex, E extends Edge<V>>
    extends AbstractGraphPathSearch<V, E> {

    //отримання значень вагових матриць на основі статистики контролера
    @LinkStats(metric = "H")
    private Map<Integer, Long> hopsCount;
    @LinkStats(metric = "B")
    private Map<Integer, Long> bandwidth;
    @LinkStats(metric = "D")
    private Map<Integer, Long> delay;
    @LinkStats(metric = "P")
    private Map<Integer, Long> packetLoss;

    //отримання значень коефіцієнтів пріоритезації
    @ValueCoefficients("priority_coefficients")
    private Map<String, Long> coefficients;

    @Override
    protected Result<V, E> internalSearch(Graph<V, E> graph, V src, V dst,
        EdgeWeigher<V, E> weigher, int maxPaths) {

        DefaultResult result = new DefaultResult(src, dst, maxPaths);

        //оновлення вагів ребер згідно коефіцієнтів пріоритезації метрик якості
        обслуговування
        updateWeightsAccordingToMetrics(weigher, coefficients);
    }
}

```

```
result.updateVertex(src, null, weigher.getInitialWeight(), false);
```

```
if (graph.getEdges().isEmpty()) {
    result.buildPaths();
    return result;
}
```

```
Heap<V> minQueue = createMinQueue(graph.getVertexes(),
    new PathCostComparator(result));
```

```
while (!minQueue.isEmpty()) {
    V nearest = minQueue.extractExtreme();
    if (nearest.equals(dst)) {
        break;
    }
}
```

```
if (result.hasCost(nearest)) {
    Weight cost = result.cost(nearest);
```

```
    for (E e : graph.getEdgesFrom(nearest)) {
        result.relaxEdge(e, cost, weigher, true);
    }
}
```

```
minQueue.heapify();
}
```

```
result.buildPaths();
```

```
return result;
```

```
}
```

```
private final class PathCostComparator implements Comparator<V> {
```

```

private final DefaultResult result;

private PathCostComparator(DefaultResult result) {
    this.result = result;
}

@Override
public int compare(V v1, V v2) {
    if (!result.hasCost(v1) && !result.hasCost(v2)) {
        return 0;
    } else if (!result.hasCost(v1)) {
        return -1;
    } else if (!result.hasCost(v2)) {
        return 1;
    }

    return result.cost(v2).compareTo(result.cost(v1));
}
}

private Heap<V> createMinQueue(Set<V> vertexes, Comparator<V> comparator) {
    return new Heap<>(new ArrayList<>(vertexes), comparator);
}

//комбінація вагових матриць для обчислення інтегрального критерію
private EdgeWeigher<V, E> updateWeightsAccordingToMetrics(EdgeWeigher<V,
E> weigher, Map<String, Long> coefficients) {
    return MatricesUtil.execute("sum",
        coefficients.get("H") * hopsCount +
        coefficients.get("B") * bandwidth +

```

```

        coefficients.get("D") * delay +
        coefficients.get("P") * packetLoss);
    }
}

```

```

public class PointToPointIntent {

    private final FilteredConnectPoint ingressPoint;
    private final FilteredConnectPoint egressPoint;

    private final List<Link> suggestedPath;

    public static PointToPointIntent.Builder builder() {
        return new Builder();
    }

    public static final class Builder extends ConnectivityIntent.Builder {
        FilteredConnectPoint ingressPoint;
        FilteredConnectPoint egressPoint;

        List<Link> suggestedPath;

        @Override
        public Builder appId(ApplicationId appId) {
            return (Builder) super.appId(appId);
        }

        @Override
        public Builder key(Key key) {
            return (Builder) super.key(key);
        }
    }
}

```



```
}
```

```
@Override
```

```
public Builder selector(TrafficSelector selector) {
    return (Builder) super.selector(selector);
}
```

```
@Override
```

```
public Builder constraints(List<Constraint> constraints) {
    return (Builder) super.constraints(constraints);
}
```

```
//калькуляція пріоритету наміру на основі оберненої ваги зв'язку
```

```
@Override
```

```
public Builder priority(List<Long> priorities) {
    Long priority = (Long) (Collections.max(priorities) + 1) - 1 /
(egressPoint.connectPoint().links(ingressPoint));
    return (Builder) super.priority(priority);
}
```

```
public Builder filteredIngressPoint(FilteredConnectPoint ingressPoint) {
    this.ingressPoint = ingressPoint;
    return this;
}
```

```
public Builder filteredEgressPoint(FilteredConnectPoint egressPoint) {
    this.egressPoint = egressPoint;
    return this;
}
```

```

public Builder suggestedPath(List<Link> links) {
    this.suggestedPath = links;
    return this;
}

//побудова наміру та включення його у path (канал зв'язку)
public PointToPointIntent build() {
    if (suggestedPath != null &&
        suggestedPath.size() > 0 &&
        (!suggestedPath.get(0)
            .src()
            .deviceId()
            .equals(ingressPoint.connectPoint().deviceId()) ||
            !suggestedPath.get(suggestedPath.size() - 1)
            .dst()
            .deviceId()
            .equals(egressPoint.connectPoint().deviceId())))
    ) {
        throw new IllegalArgumentException(
            "Suggested path not compatible with ingress and egress connect
points");
    }
    return new PointToPointIntent(
        appId,
        key,
        selector,
        ingressPoint,
        egressPoint,
        constraints,
        priority,

```

```

        suggestedPath
    );
}
}

//super-builder для наміру, що використовується при побудові каналу зв'язку
private PointToPointIntent(ApplicationId appId,
    Key key,
    TrafficSelector selector,
    FilteredConnectPoint ingressPoint,
    FilteredConnectPoint egressPoint,
    List<Constraint> constraints,
    int priority,
    List<Link> suggestedPath) {
    super(appId,
        key,
        suggestedPath != null ? resources(suggestedPath) : Collections.emptyList(),
        selector,
        constraints,
        priority);

    checkArgument(!ingressPoint.equals(egressPoint),
        "ingress and egress should be different (ingress: %s, egress: %s)",
        ingressPoint, egressPoint);

    this.ingressPoint = checkNotNull(ingressPoint);
    this.egressPoint = checkNotNull(egressPoint);
    this.suggestedPath = suggestedPath;
}
}

```

## ДОДАТОК Б

### Список публікацій здобувача

*Наукові праці, в яких опубліковано основні наукові результати дисертації:*

1. D. Korenko, O. Cherevatenko, V. Rusinov, and Y. Kulakov, “Creation of the method of multipath routing using known paths in software-defined networks,” *Technology audit and production reserves*, vol. 4, no. 2(66), pp. 19–24, Aug. 2022, doi: <https://doi.org/10.15587/2706-5448.2022.262787>.
2. Ю. Кулаков та О. Череватенко, «Моделювання маршрутизації з використанням відомих маршрутів за допомогою SDN-контролера ONOS,» *Проблеми інформатизації та управління*, 2, № 74, сс. 55–61, 2023, doi: <https://doi.org/10.18372/2073-4751.74.17882>.
3. A. Volokyta et al., “Fault Tolerance Exploration and SDN Implementation for de Bruijn Topology based on betweenness Coefficient,” *International journal of computer network and information security*, vol. 16, no. 1, pp. 97–112, Feb. 2024, doi: <https://doi.org/10.5815/ijcnis.2024.01.08>.
4. O. Cherevatenko and Y. Kulakov, “Method of Increasing Data Transmission Stability in Software Defined Network Considering Metrics of Quality of Service,” *Information, Computing and Intelligent systems*, no. 4, pp. 37–47, 2024, doi: <https://doi.org/10.20535/2786-8729.4.2024.303467>.
5. A. Volokyta, A. Kogan, O. Cherevatenko, D. Korenko, D. Oboznyi, and Y. Kulakov, “Traffic Engineering with Specified Quality of Service Parameters in Software-defined Networks,” *International Journal of Computer Network and Information Security*, vol. 16, no. 5, pp. 1–13, Oct. 2024, doi: <https://doi.org/10.5815/ijcnis.2024.05.01>.

*Апробація наукових результатів дисертації:*

6. Ю. Кулаков та О. Череватенко, «Динамічна реконфігурація комп'ютерних мереж на основі технології SDN,» *Збірник тез доповідей XV Міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології», 25–26 квітня 2024 р.,* сс. 104–105, 2024. [Online] Доступ: <https://csnt.nau.edu.ua/files/2024/sbirnyk2024.pdf>
7. O. Cherevatenko, Y. Kulakov, “Increasing the data transmission speed considering QoS parameters in SDN networks under the ONOS controller management.” *The International Conference on Security, Fault Tolerance, Intelligence*, Kyiv, Ukraine, June 28, 2024. [Online] Available at: <https://icsfti-proc.kpi.ua/issue/view/18065>.

*Праці, які додатково відображають результати дисертації:*

8. О. Гончаренко та О. Череватенко, «Способи мультиканальної маршрутизації в мережах надлишкового Де Бруйна,» *Технічні науки і технології*, № 2(24), сс. 123–130, 2021, doi: [https://doi.org/10.25140/2411-5363-2021-2\(24\)-123-130](https://doi.org/10.25140/2411-5363-2021-2(24)-123-130).