МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Кваліфікаційна наукова праця на правах рукопису

КУЗЬМІЧ МИХАЙЛО ЮРІЙОВИЧ

УДК 004:005

ДИСЕРТАЦІЯ
МЕТОДИКА ПОБУДОВИ ТА ВИКОРИСТАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ НА БАЗІ KUBERNETES ТА KUBEFLOW ДЛЯ МОБІЛЬНИХ АГЕНТІВ

Спеціальність: 123 – «Комп’ютерна інженерія»
Галузь знань: 12 – «Інформаційні технології»

Подається на здобуття наукового ступеня доктора філософії (PhD)

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

_____ Кузьміч М.Ю.

Науковий керівник: **Гордієнко Тетяна Богданівна**, доктор технічних наук, професор

Електронний підпис
Кузьміч Михайло Юрійович
Ідентифікаційний код 3496012035
Підписано у Вчасно

Київ – 2024

АНОТАЦІЯ

Кузьміч М. Ю. Методика побудови та використання моделей машинного навчання на базі Kubernetes та Kubeflow для мобільних агентів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – «Комп'ютерна інженерія» (12 – «Інформаційні технології»). – Державний університет інформаційно-комунікаційних технологій, Київ, 2024.

Дисертація присвячена розв'язанню актуального науково-технічного завдання розроблення методики побудови та використання моделей машинного навчання (МН, Mashine Learning) для мобільних агентів, якими виступають безпілотні літальні апарати (БпЛА), за допомогою автоматизованого конвеєра на основі Kubeflow із використанням кращих практик концепції МН та процесів MLOps (Machine Learning and Operation). Розроблена стратегія використання цих моделей для мобільних агентів із використанням Mesh-мереж, що є частиною описаного конвеєру.

Розвиток сучасних інформаційних технологій як науки про дані, аналітика даних і МН стають одними із головних інструментів для розв'язання складних прикладних задач у різних сферах діяльності. МН є одним із методів функціонування штучного інтелекту (ШІ), зокрема практичної реалізації його можливостей шляхом створення алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальше використання для самонавчання. Особливо це питання є актуальним в умовах дослідження війни БпЛА за допомогою ШІ як варіанту надання військовим тактичної переваги над противником та зменшення ризику для людських життів. Використовуючи МН з алгоритмами ШІ, БпЛА можуть швидко ідентифікувати загрози, реагувати на них, вести спостереження та розвідку. Зважаючи на зазначене, проведення досліджень щодо розробки нового методу із застосуванням інтегрованих МН і ШІ в керування тактикою бою БпЛА, стійкого до радіоелектронних перешкод на сьогодні є актуальним завданням.

Для досягнення поставленої в роботі мети потрібно вирішити наступні окремі завдання дослідження:

1. Проаналізувати сучасний стан розвитку МН і ШІ з можливістю їх застосування в БпЛА.

2. Дослідити можливості використання сучасних MLOps рішень для покращення процесів розробки інформаційних систем МН в розрізі інструментів Kubeflow.

3. Розробити модель побудови та застосування МН із можливістю постійного донавчання, що досягається високим рівнем інтеграції та автоматизації конвеєру із використанням компонентів Kubeflow і платформи Kubernetes.

4. Експериментально перевірити результати експлуатації моделі в арсеналі Kubeflow із використанням таких факторів покращення як швидкість розробки, імплементації змін, зменшення часу на пошук проблем, відновлення після глобальних перебоїв, зменшення кількості помилок в моделі.

5. Розробити архітектурну концепцію системи на базі дистрибутива Kubernetes k3s з використанням моделі машинного навчання в парадигмі граничних обчислень (Edge computing) для мобільних агентів із використанням mesh-мереж.

6. Провести оцінку ефективності використання бездротової Mesh-мережі для підвищення функціональної стійкості розподіленої інформаційної системи БпЛА.

Об’єктом дослідження є процес розгортання конвеєра розробки та використання моделей МН.

Предметом дослідження є методи та алгоритми розробки конвеєра на основі платформи Kubernetes та інструментів Kubeflow із специфікою впровадження для мобільних агентів.

Методи дослідження. Проведені дослідження базуються на методах прийняття рішень, методах ШІ, комп’ютерного та об’єктно-орієнтованого програмування.

Метою дослідження є розробка моделей і методів покращення процесів розробки інформаційних систем МН з використанням сучасних MLOps рішень, таких як Kubeflow, для мобільних агентів.

Вірогідність наукових результатів, висновків та рекомендацій, викладених у дисертаційній роботі, обґрунтовано моделюванням на персональному комп'ютері, збігом результатів теоретичних досліджень з результатами моделювання.

Наукова новизна одержаних результатів. У процесі теоретичних і експериментальних досліджень та моделювання одержано наступні нові наукові результати:

1. Вперше сформовано концепцію повноцінного інформаційного рішення на базі конвеєру неперервної інтеграції із можливістю постійного донавчання, високого рівня інтеграції та автоматизації з використанням компонентів Kubeflow і платформи Kubernetes, що дозволяє покращити кількісну та якісну складові експериментів, зменшити час, затрачений на його підготовку, та мінімізувати помилки, спричинені людським фактором.

2. Вперше спроектовано архітектурну концепцію системи на базі дистрибутива Kubernetes k3s, що дозволяє ефективно використовувати моделі МН в парадигмі граничних обчислень (Edge Computing) із використанням Mesh-мереж для мобільних агентів (БпЛА), що формують собою функціонально стійку розподілену інформаційну систему.

3. Удосконалено модель підвищення функціональної стійкості розподіленої інформаційної системи БпЛА з використанням бездротової Mesh-мережі передачі даних, що дозволить протистояти радіоелектронним та іншим перешкодам і працювати в автономному або напів-автономному режимі.

Практичне значення одержаних результатів у галузі розробки та створення ефективних методів побудови та використання моделей МН для БпЛА полягає в наступному:

1. Запропоновані архітектурні рішення інформаційних систем та метод можуть бути використані дослідно-конструкторськими організаціями та

державними структурами Збройних сил України для реалізації концепції “Аналітична система підтримки прийняття рішень для БпЛА“, що показано через функціональні моделі процесів (AS, TO-BE, SHOULD-BE).

2. Сучасні MLOps рішень на базі інструментів Kubeflow для створення та використання навчальних моделей МН впроваджено в дисципліну “Системи підтримки прийняття рішень”. Високий рівень автоматизації та інтеграції компонентів дозволяє створити та запустити тестову модель за відносно короткий час, але без зниження якості її роботи та надійності.

Дисертаційна робота складається з вступу, чотирьох розділів, в яких викладено рішення поставлених задач дослідження.

У вступі обґрунтовано актуальність теми дисертаційного дослідження, сформульовано мету дослідження та науково-технічні завдання, необхідні для її досягнення, показано зв’язок дослідження з науковими програмами та темами, наведено наукову новизну отриманих результатів, їх практичну цінність та особистий внесок здобувача, надано інформацію про апробацію результатів роботи.

В першому розділі проведено аналіз сучасного стану розвитку основних типів і алгоритмів МН і ШІ з можливістю їх застосування в БпЛА. Проведено аналіз концепції MLOps, що є набором технік для імплементації та автоматичної неперервної інтеграції, а також доставки на продуктове середовище та навчання моделей. Ефективність МН визначають такі чинники, як доступ до великих наборів даних, невисока ціна обчислювальних ресурсів, функціональні сервіси на основі хмарних технологій, швидкий прогрес у сферах ШІ, такі як комп’ютерний зір, розуміння природної мови та рекомендаційні системи. Разом із тим у розробках МН існує велика кількість додаткових складнощів, через що безліч пов’язаних систем зазнає невдачі на ранніх стадіях розробки і не досягає продуктового середовища. Також розглянуто використання, імплементацію та автоматизацію розгортання за протоколом технології Web комунікації в реальному часі WebRTC (Web real-time communications) шлюзу вибіркового направлення у хмарному середовищі (Cloud Native).

У другому розділі розглянуто концепцію MLOps в розрізі інструментів Kubeflow – хмарних систем з відкритим кодом, що працює на платформі Kubernetes. Завдання останнього не дублювати інші служби, а забезпечити простий спосіб розгортання найкращих систем з відкритим кодом для МН на різних інфраструктурах, будь то приватні сервери на виробництві або публічні хмарні рішення. Тобто там, де працює Kubernetes, може працювати Kubeflow, що робить його гнучким та портативним. Проведені дослідження показали, що Kubeflow складається із набору різноманітних компонентів відкритих джерел (open source), що мають високий рівень інтеграції між собою через платформу Kubernetes. При цьому Kubeflow надзвичайно ефективно використовує патерн Kubernetes операторів для об’єктів МН.

Один із важливих етапів конвеєра МН це тренування моделі. Під час цього процесу алгоритм МН подається даними, з яких він може навчатися. Потенційне перенавчання моделі можна виконувати на регулярній основі щоб адаптуватись до змін та постійно покращувати ефективність. Оброблені відео, на основі яких модель показала низьку ефективність, потрібно проаналізувати та обробити оператором для перенавчання моделі. За рахунок цього модель буде постійно покращувати свою ефективність та швидко адаптуватись під зміни навколишнього середовища.

Актуальність застосування Mesh-мереж визначається появою безлічі різних пристроїв, здатних працювати автономно довгий час із багаторазовою зміною режиму (Online находження в мережі і Offline-виходу з мережі) і потребою обміну інформацією зі своїм оточенням, а можливо і з керуючим або інформаційним центром. Одна з переваг Mesh-мереж – незалежність та стійкість. За задумом, у повноцінних Mesh-мережах неможливо перехопити трафік і заборонити поширення інформації. Через ці причини державні структури та військові відомства зацікавлені в освоєнні і організації подібних мереж.

У третьому розділі зроблено акцент на пошуку оптимального способу розгортання та наведено приклад декларативного опису розгортання SFU (Selective Forwarding Unit) шлюзу на сервері Jitsi у вигляді коду, що може бути

використаний для наступних експериментів. Встановлено, що розгортання та експлуатація SFU серверів в Cloud Native є можливим та значно ефективнішим при використанні переваг платформи Kubernetes, що відкриває можливості для подальших досліджень в цьому напрямку. Процес розгортання було автоматизовано з використанням Helm – пакетного менеджера для Kubernetes. Рекомендації з покращення Helm пакета, які сформовано при експлуатації, було доведено до розробника через платформу Github.

Можливості сучасних MLOps рішень можуть сприяти покращенню процесів розробки інформаційних систем МН, зокрема з алгоритмами ШІ. В роботі спроектовано інформаційну систему на базі МН із можливістю постійного навчання, що досягається високим рівнем інтеграції та автоматизації конвеєру. Запропоновано та обґрунтовано використання Mesh-мереж для підвищення функціональної стійкості розподіленої інформаційно системи мобільних агентів.

Спроектовано архітектурну концепцією системи на базі дистрибутива Kubernetes k3s, що дозволяєте ефективно використовувати моделі МН в парадигмі граничних (Edge) обчислень. З огляду на високу портативність додатків, розроблених в Kuberntes та Kubeflow, проаналізовано можливість розгортання моделей МН на мобільних агентах. Водночас, доведено, що наявність повноцінного конвеєра для неперервної інтеграції та постачання додатків є ключовою для успішної реалізації цього процесу від початкової розробки до кінцевого користувача.

Оскільки Kubernetes початково не був розроблений для роботи із бездротовими стандартами передачі даних, в тому числі Mesh-мережами, на сьогодні є дистрибутиви саме для таких задач: KubeEdge, k3s, MicroK8s. Порівнюючи ці платформи, для структурної схеми інформаційної системи обрано k3s, який користується більшою популярністю та виграє по ряду характеристик.

Четвертий розділ присвячено впровадженню та експериментальній перевірці результатів експлуатації моделі в арсеналі Kubeflow. На основі проведено аналізу основних дистрибутив Kubernetes, що використовуються для

Edge Computing, обґрунтована переваги та доцільність використання k3s для поточної системи. Наведена концепція використання MLOps конвеєру для розв'язання прикладної задачі класифікації об'єктів із відео розвідувальних БпЛА є першою спробою представити комплексну систему МН як цілісний об'єкт, що може формувати додаткову цінність для продукту.

На основі проведених досліджень було сформовано концепцію інформаційного рішення на основі ШІ для виявлення ворожих цілей із БПлА. Для підвищення функціональної стійкості розподіленої інформаційної системи БПлА розглянуто використання бездротової Mesh-мережі, протоколи передачі якої регулюються стандартами Інституту інженерів електротехніки та електроніки IEEE 802.11s.

Топологія Mesh-мережі адаптивна та спрямована на використання будь-яких доступних каналів для забезпечення повної маршрутизації між усіма учасниками мережі. Це робить її надзвичайно стійкою до радіоелектронних перешкод. Основна ідея полягає в можливості забезпечити функціонально стійку мережу передачі даних для поточної інформаційної системи на основі БпЛА, яка зможе протистояти сучасним засобам радіоелектронних перешкод та іншим перешкодам і працювати в автономному або напів-автономному режимі.

Запропоновано використання Mesh-мереж для БпЛА, які представляють собою розподілену інформаційну систему на базі дистрибутива Kubernetes k3s та Kubeflow, що працюють в парадигмі Edge Computing. Доведено, що використання Mesh-мереж дозволяє досягти високої функціональної стійкості інформаційної системи БПлА в умовах протистояння радіоелектронним та іншим перешкодам і роботи в автономному або напів-автономному режимі.

Як і Kubernetes, k3s має схожу архітектуру розділення на службові ноди (Server) та ноди працівники (Agent). На Server ноди встановлені необхідні службові компоненти для роботи кластеру та Agent-компонети, що дозволяють їм запускати клієнтські застосунки також. На Agent-ноди службові компоненти не встановлюються, тільки базові. У такій системі доцільно мати хоча б три

БпЛА управління, для досягнення високої доступності та можливості розподілення навантаження між вузлами.

Ключові слова: розподілені інформаційні системи, машинне навчання, штучний інтелект, мобільні агенти, Mesh-мережі, безпілотний літальний апарат, MLOps, Kuberentes, Kubeflow, граничні обчислення, розгортання тестової моделі.

ABSTRACT

Kuzmich M. Yu. Methodology for building and using Machine Learning models based on Kubernetes and Kubeflow for mobile agents. – Qualifying scientific work on manuscript rights. – Qualifying scientific work on the manuscript.

The thesis for the Philosophy Doctor (Ph.D.) degree in specialty 123 – "Computer Engineering" (12 – "Information Technology"). – State University of Information and Communication Technologies, Kyiv, 2024.

The dissertation is devoted to the solution of the current scientific and technical task of developing a methodology for building and using machine learning (ML) models for mobile agents that act as unmanned aerial vehicles (UAVs), using an automated pipeline based on Kubeflow using the best practices of the ML concept and MLOps (Machine Learning and Operation) processes. A strategy for using these models for mobile agents using Mesh networks, which is part of the described pipeline, is developed.

The development of modern information technologies as data science, data analytics and ML are becoming one of the main tools for solving complex applied problems in various spheres of activity. ML is one of the methods of Artificial Intelligence (AI), in particular, the practical implementation of its capabilities by creating algorithms for identifying patterns during the analysis of big data, and their further use for self-learning. This issue is especially relevant in the context of the study of UAV warfare with the help of AI as an option for giving the military a tactical advantage over the enemy and reducing the risk to human lives. Using ML with AI algorithms, UAVs can quickly identify threats, respond to them, conduct surveillance and reconnaissance. Considering the above, conducting research on the development of a new method with the use of integrated ML and AI in controlling the combat tactics of UAVs that is resistant to radio-electronic interference is currently an urgent task.

To achieve the goal set in the work, it is necessary to solve the following separate research tasks:

1. To analyze the current state of development of ML and AI with the possibility of their application in UAVs.

2. Investigate the possibilities of using modern MLOps solutions to improve the development processes of information systems of the ML in the context of Kubeflow tools.

3. Develop a model for building and applying ML with the possibility of continuous training, which is achieved by a high level of integration and automation of the pipeline using Kubeflow components and the Kubernetes platform.

4. Experimentally check the results of operating the model in the Kubeflow arsenal using such improvement factors as speed of development, implementation of changes, reduction of time to search for problems, recovery after global interruptions, reduction of the number of errors in the model.

5. Develop an architectural concept of the system based on the Kubernetes k3s distribution using a machine learning model in the edge computing paradigm for mobile agents using mesh networks.

6. To evaluate the effectiveness of using a wireless Mesh network to increase the functional stability of the distributed information system of the of the UAV.

The object of the research is the process of deployment of the pipeline of development and use of MH models.

The subject of the research is methods and algorithms for pipeline development based on the Kubernetes platform and Kubeflow tools with implementation specificity for mobile agents.

Research methods. The conducted research is based on decision-making methods, methods of AI, computer and object-oriented programming.

The purpose of the research is to develop models and methods for improving the development processes of information systems of the UAV using modern MLOps solutions, such as Kubeflow, for mobile agents.

The plausibility of the scientific results, conclusions and recommendations presented in the dissertation is substantiated by simulation on a personal computer, the coincidence of the results of theoretical research with the results of simulation.

Scientific novelty of the obtained results. In the process of theoretical and experimental research and modeling, the following new scientific results were obtained:

1. For the first time, the concept of a full-fledged information solution based on a continuous integration pipeline with the possibility of continuous training, a high level of integration and automation using Kubeflow components and the Kubernetes platform was formed, which allows to improve the quantitative and qualitative components of experiments, reduce the time spent on its preparation, and minimize errors caused by the human factor.

2. For the first time, the architectural concept of the system based on the Kubernetes k3s distribution was designed, which allows the effective use of ML models in the paradigm of edge computing (Edge Computing) with the use of Mesh-networks for mobile agents (UAVs), which form a functionally stable distributed information system.

3. The model for increasing the functional stability of the distributed information system of UAVs using a wireless Mesh-network of data transmission has been improved, which will allow to resist radio-electronic and other interferences and work in autonomous or semi-autonomous mode.

The practical significance of the obtained results in the field of development and creation of effective methods of building and using ML models for UAVs is as follows:

1. The proposed architectural solutions of information systems and the method can be used by research and development organizations and state structures of the Armed Forces of Ukraine to implement the concept “Analytical decision support system for UAVs”, which is shown through functional process models (AS, TO-BE and SHOULD- BE).

2. Modern MLOps solutions based on Kubeflow tools for the creation and use of educational models of ML are implemented in the discipline “Decision support systems”. A high level of automation and integration of components allows you to create and run a test model in a relatively short time, but without reducing the quality of its work and reliability.

The dissertation work consists of an introduction, four chapters, in which the solutions to the research problems are outlined.

The introduction substantiates the relevance of the topic of the dissertation research, formulates the purpose of the research and the scientific and technical tasks necessary for its achievement, shows the connection of the research with scientific programs and topics, provides the scientific novelty of the obtained results, their practical value and the personal contribution of the recipient, provides information about approval of work results.

In the first section, an analysis of the current state of development of the main types and algorithms of ML and AI with the possibility of their application in UAVs was carried out. An analysis of the concept of MLOps, which is a set of techniques for implementation and automatic continuous integration, as well as delivery to the production environment and training of models, was carried out. The effectiveness of ML is determined by such factors as access to large data sets, low cost of computing resources, functional services based on cloud technologies, rapid progress in the fields of AI, such as computer vision, natural language understanding, and recommender systems. However, there are a many number of additional complexities in ML development, due to which many related systems fail in the early stages of development and do not reach the product environment. The use, implementation, and automation of deployment using the Web RTC (Web real-time communications) technology protocol of the selective forwarding gateway in the Cloud Native are also considered.

In the second section, the concept of MLOps is considered in terms of Kubeflow tools – open source cloud systems running on the Kubernetes platform. The task of the latter is not to duplicate other services, but to provide an easy way to deploy the best open source systems for ML on different infrastructures, be it private servers in production or public cloud solutions. That is, where Kubernetes runs, Kubeflow can run, making it flexible and portable. The conducted studies showed that Kubeflow consists of a set of various open source components that have a high level of integration with each other through the Kubernetes platform. At the same time, Kubeflow uses the Kubernetes pattern of operators for ML objects extremely efficiently.

One of the important stages of the ML pipeline is model training. During this process, the ML algorithm is provided with data from which it can learn. Potential retraining of the model can be performed on a regular basis to adapt to changes and continuously improve performance. Processed videos based on which the model showed poor performance need to be analyzed and processed by the operator to retrain the model. Due to this, the model will constantly improve its efficiency and quickly adapt to changes in the environment.

The relevance of the use of Mesh networks is determined by the appearance of many different devices that can work autonomously for a long time with multiple mode changes (online presence in the network and offline exit from the network) and the need to exchange information with their environment, and possibly with the control or information center. One of the advantages of Mesh networks is independence and stability. By design, it is impossible to intercept traffic and prohibit the dissemination of information in full-fledged Mesh networks. For these reasons, state structures and military departments are interested in the development and organization of such networks.

The third section focuses on finding the optimal deployment method and provides an example of a declarative description of the SFU (Selective Forwarding Unit) deployment of the gateway on the Jitsi server in the form of code that can be used for further experiments. It has been established that the deployment and operation of SFU servers in Cloud Native is possible and significantly more efficient when using the advantages of the Kubernetes platform, which open opportunities for further research in this direction. The deployment process was automated using Helm, a package manager for Kubernetes. Recommendations for improving the Helm package, which were generated during operation, were brought to the developer through the Github platform.

The capabilities of modern MLOps solutions can contribute to the improvement of the development processes of ML information systems, with AI algorithms. In the work, an information system based on ML was designed with the possibility of continuous learning, which is achieved by a high level of integration and automation

of the conveyor. The use of Mesh networks to increase the functional stability of the distributed information system of mobile agents is proposed and substantiated.

It is designed with the architectural concept of the system based on the Kubernetes k3s distribution, which allows you to effectively use ML models in the edge computing paradigm. Given the high portability of applications developed in Kuberntes and Kubeflow, the possibility of deploying ML models on mobile agents was analyzed. At the same time, it has been proven that having a full-fledged pipeline for continuous integration and delivery of applications is key to successfully implementing this process from initial development to the end user.

Since Kubernetes was not originally designed to work with wireless data transmission standards, including Mesh networks, today there are distributions for such tasks: KubeEdge, k3s, MicroK8s. Comparing these platforms, k3s is chosen for the structural diagram of the information system, which is more popular and wins in a number of characteristics.

The fourth section is devoted to the implementation and experimental verification of the results of operating the model in the Kubeflow arsenal. Based on an analysis of the main Kubernetes distributions used for Edge Computing, the advantages and feasibility of using k3s for the current system are justified. The presented concept of using the MLOps pipeline to solve the applied problem of classifying objects from the video of reconnaissance UAVs is the first attempt to present a complex ML system as a complete object that can form additional value for the product.

Based on the conducted research, the concept of an AI-based information solution for detecting enemy targets with air defense systems was formed. In order to increase the functional stability of the distributed information system of the Federal Aviation Administration, the use of a wireless Mesh network, the transmission protocols of which are regulated by the standard IEEE 802.11s of the Institute of Electrical and Electronics Engineers, is considered.

Mesh network topology is adaptive and aims to use any available channels to ensure complete routing between all network participants. This makes it extremely resistant to radio-electronic interference. The main idea is to provide a functionally

stable data transmission network for the current UAV-based information system that can withstand modern means of radio-electronic jamming and other interference and work in an autonomous or semi-autonomous mode.

The use of Mesh networks for UAVs, which are a distributed information system based on the Kubernetes k3s and Kubeflow distribution, operating in the Edge Computing paradigm, is proposed. It has been proven that the use of Mesh-networks allows to achieve high functional stability of the information system of the UAVs in the conditions of resistance to radio-electronic and other interferences and operation in autonomous or semi-autonomous mode.

Like Kubernetes, k3s has a similar architecture of separation into service nodes (Server) and worker nodes (Agent). The necessary service components for the operation of the cluster and Agent components are installed on the Server nodes, which allow them to run client applications as well. Service components are not installed on Agent nodes, only basic ones. In such a system, it is advisable to have at least three management control units to achieve high availability and the ability to distribute the load between nodes.

Key words: distributed information systems, machine learning, artificial intelligence, mobile agents, Mesh networks, unmanned aerial vehicle, MLOps, Kuberentes, Kubeflow, Edge Computing, test model deployment.

Список публікацій за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. M. Kuzmich, T. Gordiyenko, “Application of Kubeflow as a universal approach for the development and implementation of artificial intelligence systems”, ARPN Journal of Engineering and Applied Sciences, vol. 18, 2023, no. 20, pp. 2311–2320 (**Scopus**).

2. Кузьміч М. Ю., Гордієнко Т. Б., “Застосування інструменту Kubeflow для інтеграції машинного навчання і штучного інтелекту в безпілотних літальних апаратах”. Телекомунікаційні та інформаційні технології, 2023, вип. 3 (80), с. 65–78.

3. Кузьміч М. Ю., Гордієнко Т. Б., “Імплементація та автоматизація розгортання WebRTC застосунків в Cloud Native оточенні”. Телекомунікаційні та інформаційні технології, 2021, вип. 3 (72), с. 54–62.

4. Кузьміч М. Ю., Гордієнко Т. Б., “Упровадження Kubeflow MLOps у розподілені інформаційні системи мобільних агентів із покращеною функціональною стійкістю”. Зв’язок, 2023, № 6 (166), с. 28–32.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

5. Kuzmich M. Ю., Zolotuhina O. A., “Analysis of challenges faced by implementation WebRTC SFU gateway in cluster computing environment”. Збірник матеріалів Науково-технічної конференції “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”, Київ: ДУТ, 2020, с. 58–60.

6. Кузьміч М. Ю., “Особливості імплементація WebRTC застосунків в cloud native оточенні”. Збірник матеріалів науково-практичної конференцію “Проблеми комп’ютерної інженерії”, Київ: ДУТ, 2020, с. 23–25.

7. Кузьміч М. Ю., Кравчук С. О., “Дослідження мобільної Mesh-мережі безпілотних літальних апаратів з урахуванням затримки між вузлами”. Збірник матеріалів Міжнародної науково-технічної конференції “Перспективи

телекомунікацій”, Київ: Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, 2017, с. 63–65.

Наукові праці, які додатково відображають наукові результати дисертації:

8. Кузьміч М. Ю., Павлов А. О., “Інформаційна система ідентифікації користувачів на основі біометричних даних”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 39–43.

9. Кузьміч М. Ю., Костенко В. К., “Інформаційна система управління проектами”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 106–108.

10. Кузьміч М. Ю., Дубовой Д. В., “Розробка інформаційної системи рекомендування ціни вживаного авто”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 194–198.

11. Кузьміч М. Ю., Сміхун О. Ф., “Інформаційна система аналізу зміни стану рослин сільськогосподарських угідь з урахуванням даних їх дистанційного зондування”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 111–116.

ЗМІСТ

АНОТАЦІЯ …… 2

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ …… 23

ВСТУП ……..…… 27

РОЗДІЛ 1. АНАЛІЗ РОЗВИТКУ ОСНОВНИХ ТИПІВ МАШИННОГО НАВЧАННЯ …….… 35

1.1 Розвиток і сфери застосування машинного навчання …… 35

1.2 Основні типи машинного навчання …… 37

1.3 Різновиди машинного навчання …….….. 41

1.3.1 Кероване і некероване машинне навчання ..…….…….. 41

1.3.2 Глибинне навчання ……... 43

1.4 Концепція спільного підходу до процесів розробки та експлуатації моделей машинного навчання …….…. 46

1.4.1 Розвиток моделей машинного навчання та експлуатації …….. 46

1.4.2 Аналіз сучасних досліджень у галузі розробки та експлуатації моделей машинного навчання ……..……... 48

1.5 Автоматизація розгортання на прикладі WebRTC застосунків …… 52

1.6 Цілі та задачі дисертаційного дослідження ……. 55

1.7 Список використаних джерел ……... 56

РОЗДІЛ 2. ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ Kubernetes ТА Kubeflow У КОНТЕКСТІ MLOps …… 63

2.1. Особливості застосування технології Kubernetes …… 63

2.2 Основні характеристики і компоненти Kubernetes ……. 67

2.2.1 Основні характеристики Kubernetes …… 67

2.2.2 Основні компоненти Kubernetes …….. 69

2.3 Контролери Kubernetes …….. 71
2.3 Оператори Kubernetes …… 76
2.4 Методи дослідження роботи конвеєра машинного навчання ……... 79
2.5 Моделювання у Kubeflow із застосування Seldon Core Serving …… 85
2.6 Висновки до розділу …….. 88
2.7 Список використаних джерел ……... 90

РОЗДІЛ 3. ВИКОРИСТАННЯ СИСТЕМИ РОЗПОДІЛЕНОЇ ОБРОБКИ ДАНИХ НА ОСНОВІ MESH-МЕРЕЖІ ТА ЇЇ ІНТЕГРАЦІЯ З ПОТОЧНОЮ ІНФРАСТРУКТУРОЮ …….….. 91
3.1 Основи побудови Mesh-мережі …… 91
3.2 Mesh-мережа за стандартом IEEE 802.11s …….……. 99
3.2.1 Загальні характеристики Mesh-мережі …….…… 99
3.2.2 Підключення WLAN Mesh …….…….. 101
3.3 Пропускна здатність і менеджмент ресурсів у Mesh-мережі …….. 105
3.4 Проблемні питання впровадження Mesh-мереж …….…….. 108
3.4.1 Особливості мережевих архітектур …….…….. 108
3.4.2. Проблеми за рівнями взаємозв’язку відкритих систем …….…. 109
3.4.3 Питання безпеки в бездротових Mesh-мережах …….….. 112
3.4.4 Mesh-сценарії користувача……..… 116
3.5 Основні дистрибутиви Kubernetes для Edge Computing ……... 119
3.6 Особливості реалізацій Mesh-мереж EdgeMesh і Mesh Rider ..……. 124
3.6.1 EdgeMesh мережа ……..... 124
3.6.2 Mesh Rider мережа ……... 126
3.7. Висновки до розділу ……..….. 127
3.8 Список використаних джерел ……..… 128

РОЗДІЛ 4. РЕЗУЛЬТАТИ ВПРОВАДЖЕННЯ ПІДХОДУ MLOps ІЗ ВИКОРИСТАННЯМ Kubeflow ТА Kubernetes …….. 131
4.1 Впровадження системи машинного навчання для мобільних агентів ……131
4.2 Етапи машинного навчання конвеєру в розрізі моделі розпізнавання об’єктів …… 134
4.3 Функціональні моделі процесів AS-IS, TO-BE і SHOULD-BE …… 135
4.3.1 Загальна характеристика функціональних моделей процесів …. 135
4.3.2 Опис функціональної моделі процесів AS-IS …… 136
4.3.3 Опис функціональної моделі процесів TO-BE …….. 138
4.3.4 Опис функціональної моделі процесів SHOULD-BE ……... 141
4.4 Використання Mesh-мереж для мобільних агентів …… 143
4.5 Функціональна стійкість Mesh-мережі на основі мобільних агентів 146
4.5.1 Особливості існуючих рішень ……. 146
4.5.2 Обґрунтування вибору компонентів мережі …….. 147
4.5.3 Вибір оптимального протоколу маршрутизації ...……. 149
4.5.4 Гібридний бездротовий Mesh-протокол маршрутизації HWMP . 151
4.5.5 Вибір архітектури з урахуванням функціональної стійкості мережі ……..153
4.6 Моделювання і симуляції функціонально-стійкої Mesh-мережі на основі безпілотних літальних апаратів …….. 155
4.7 Особливості використання хмарного провайдера AWS …… 160
4.8 Особливості використання мережевого симулятора NS-3 …… 162
4.9 Висновки до розділу …….…… 167
4.10 Список використаних джерел ……...…… 168

ВИСНОВКИ …….… 171

ДОДАТКИ …… 172
ДОДАТОК А. Лістинги розрахунково-експериментальних моделей … 174

А1. Мережевий пакету в програмі Wireshark просимульованої мережі .174
А2. Код симуляції Mesh-мережі 5x5 стандарту IEEE 802.11s ...……..... 184
ДОДАТОК Б. Список публікацій здобувача за темою дисертації …..... 186
ДОДАТОК В. Акти впровадження результатів дисертаційної роботи .. 188

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БС – базова станція (base station)

ВМ – віртуальна машина (virtual machine)

МН – машинне навчання (machine learning)

МС – мобільна станція (mobil station)

ОС – операційна система

СУБД – система управління базами даних

ШІ – штучний інтелект (Artificial Intelligence)

ШНМ – штучна нейронна мережа (Artificial Neural Network)

AAA – автентифікація, авторизація, облік (Authentication, Authorization, Accounting)

AIOps – ШІ та експлуатація (Artificial Intelligence and Operation)

ALM – метрика часу передачі в каналі (Airtime Link Metric)

AODV – дистанційно-векторна маршрутизація за запитом (Ad hoc On Demand Distance Vector)

AP – точка доступу (Access Point)

API – прикладний програмний інтерфейс (Application Programming Interface)

AS – сервер аутентифікації (Authentication Server)

AWS – Amazon Web служби (Amazon Web Services)

CDMA – множинний доступ з кодовим розділенням (Code Division Multiple Access)

CLI – інтерфейс командного рядка (Command Line Interface)

CNCF – Фонд природних обчислень (Cloud Native Computing Foundation)

CNI – мережевий інтерфейс контейнера (Container Network Interface)

CNN – згорточна нейронна мережа (Convolutional Neural Network)

CRD – спеціальне визначення ресурсів (Custom Resource Definitions)

CRI – інтерфейс виконання контейнера (Container Runtime Interface)

CSCF – централізована конфігурація (Centralized Configuration)

CSCH – централізоване планування (Centralized Scheduling)

CSMA/CA – множинний доступ із визначенням несучої з уникненням конфліктів (Carrier Sense Multiple Access With Collision Avoidance)

DCF – функція розподіленої координації (Distributed Coordination Function)

DHT – розподілена геш-таблиця (Distributed Hash Table)

DPM – модель домінантного шляху (Dominant Path Model)

DS – розподільча система (Distribution System)

DSCH – керування розподілом (Distributed Scheduling)

DSL – предметно-орієнтована мова програмування ()

EAP – розширений протокол аутентифікації (Extensible Authentication Protocol)

EAPOL – протокол або розширюваний протокол автентифікації через локальну мережу (Protocol or Extensible Authentication Protocol over LAN)

ESP – інкапсуляція корисного вантажу безпеки (Encapsulation Security Payload)

FCS – послідовності перевірки кадрів (Frame Check Sequence)

GW – шлюзовий вузол (Gataway Node)

HAP – високо розташована платформа (Hight Altitude Platform)

HTC – контроль високої пропускної здатності (High Throughput Control)

HWMP – гібридний безпровідний mesh-протокол маршрутизації (Hybrid Wireless Mesh Protocol)

IoT – інтернет речей (Internet of Things)

IP – міжмережний протокол (Internet Protocol)

IPSec – скорочення від IP Security

NAS – пошук нейронної архітектури (Neural Architecture Search)

LAN – локальна мережа (Local Area Network)

LAP – низько розташована платформа (Low Atitude Platform)

LOS – пряма видимість (Line-of-Sight)

MAC – управління доступом до середовища (Medium Access Control)

MANET – мобільна Ad hoc мережа (Mobile Ad-hoc Networks)

MAP – точка Mesh доступу (Mesh Access Point)

MCU – шлюз багатоточкового контролю (Multipoint Control Unit)

MDA – Mesh детермінований доступ (Mesh Deterministic Access)

MIMO – множинний вхід множинний вихід (Multiple Input Multiple Output)

MLife – МН життєвий жикл (Mashine Learning Life)

MLOps – МН та експлуатація (Machine Learning and Operation)

MN – вузол Mesh (Mesh Node)

MP – точка Mesh (Mesh Point)

MPP – портал Mesh-мережі (Mesh Point Portal)

MPR – багатоточкова естафета (Multi Point Relay)

MR-WMN – мультирадіо WMN (Multi Radio WMN)

MQTT – (Message Queue Telemetry Transport)

MUN – Mesh-вузли користувачів (Mesh User Node)

REST – Передача представницького стану (Representational State Transfer)

RTM – моделі трасування променів (Ray Tracing Model)

OFDM – ортогональне частотне розділення каналів мультиплексування (Orthogonal Frequency-Division Multiple Access)

RP-OLSR – оптимізований протокол маршрутизації стану зв'язку (Optimized Link State Routing Protocol)

QPSK – квадратурна фазова маніпуляція (Quadrature Phase Shift Keying)

QoS – якість обслуговування (Quality of Service)

PGP – достатньо надійна конфіденційність (Pretty Good Privacy)

RTT – час проходження маршруту (route-trip-time)

SA – асоціації безпеки (Security Association)

SDK – набор засобів розроблення (Software Development Kit)

SFU – шлюз вибіркового направлення (Selective Forwarding Unit)

STA – абонентські термінальні станції (Station Terminal Abonent)SSL – рівень захищених сокетів (Secure Sockets Layer)

TCP – протокол транспортного рівня (Transmission Control Protocol)

TKIP – протокол цілісності тимчасового ключа (Temporal Key Integrity Protocol)

TLS – захист на транспортному рівні (Transport Layer Security)

UN – вузол користувачів (User Node)

UWB – надширока смуга (Ultra-Wide Band)

VPN – віртуальна приватна мережа (Virtual Private Network)

WDS – бездротова розподільча система (Wireless Distribution System)

WebRTC – Веб-комунікації в реальному часі (Web Real-Time Communications)

WiFi – бездротова вірність відтворення (Wireless Fidelity)

WLAN – бездротова локальна мережа (Wireless Local Area Network)

WMN – бездротова mesh-мережа (Wireless Mesh Network)

WPA – захищений доступ до Wi-Fi (Wi-Fi Protected Access)

ВСТУП

Актуальність теми дисертації. Розвиток сучасних інформаційних технологій як науки про дані, аналітика даних і машинного навчання (МН) стають одними із головних інструментів для розв'язання складних прикладних задач у різних сферах діяльності. МН є одним із методів функціонування штучного інтелекту (ШІ), зокрема практичної реалізації його можливостей шляхом створення алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальше використання для самонавчання [1–3]. Швидкий прогрес у різних галузях досліджень МН (таких як комп'ютерний зір, розуміння природної мови та рекомендаційні системи ШІ) мотивують більшість компаній до інвестицій у свої команди з обробки даних і можливості МН, щоб розробити прогнозні моделі, які можуть принести бізнес-цінність своїм користувачам. Особливо це питання є актуальним в умовах дослідження війни безпілотних літальних апаратів (БпЛА) за допомогою ШІ як варіанту надання військовим тактичної переваги над противником та зменшення ризику для людських життів. Використовуючи МН з алгоритмами ШІ, БпЛА можуть швидко ідентифікувати загрози, реагувати на них, вести спостереження та розвідку.

Разом із тим у розробці МН існує велика кількість додаткових складнощів, через це безліч систем МН зазнає невдачі на ранніх стадіях розробки і не досягає продуктового середовища. Для вирішення цих проблем використовують концепцію спільного підходу до процесів МН та експлуатації MLOps (Machine Learning and Operation), яка є набором технік для імплементації та автоматичної неперервної інтеграції, а також доставки на продуктове середовище та навчання моделей. Ефективність використання підходу MLOps розглянуто і доведено в багатьох роботах, де було формалізовано основні вимоги до систем МН [4, 5]. Фахівці з обробки даних можуть реалізувати та навчити модель МН із прогнозованою ефективністю на офлайн-затриманому наборі даних, маючи відповідні навчальні дані для свого сценарію використання. Однак справжнє

завдання полягає не в побудові моделі МН, а в побудові інтегрованої системи МН і її безперервній експлуатації у виробництві, що може мати певні складнощі. Останні виявлені працівниками Google щодо невидимого технічного боргу в системах МН, оскільки для їх повноцінної роботи потрібна складна та обширна інфраструктура [6].

Тому розробники програмного забезпечення все більше віддають перевагу концепції контейнеризації. В основі концепції контейнеризації та мікросервісів розроблена розподілена система керування контейнерами з відкритим кодом під назвою Kubernetes. Останній може не лише підтримувати повну незалежність програм, але й покращувати використання апаратних ресурсів, тому його широко використовують більшість установ. Таким чином концепцію MLOps доцільно розглядати в розрізі інструментів Kubeflow – Cloud Native системи з відкритим кодом, що працює на платформі Kubernetes.

Особливо це питання є актуальним в умовах дослідження війни БпЛА за допомогою ШІ як варіанту надання військовим тактичної переваги над противником і зменшення ризику для людських життів. Використовуючи МН з алгоритмами ШІ, БпЛА можуть швидко ідентифікувати загрози, реагувати на них, вести спостереження та розвідку [7, 8]. Також для підвищення функціональної стійкості розподіленої інформаційної системи мобільних агентів (БпЛА) доцільно розглянути використання бездротових мереж на кшталт Mesh-мереж. Описано застосування Mesh-мереж для платформи Kubernetes і потенційні переваги використання оновленого компонента планувальника навантаження із можливістю оцінювати затримки між вузлами. Запропоноване рішення покращило загальну продуктивність розподіленої системи при необхідності у збільшенні кількості вузлів. Однак воно не достатньо враховує практичні переваги застосування Mesh-мереж в Kubernetes і дистрибутиви, розроблені для систем в парадигмі кордонних обчислень (Edge computing) [9].

Описана структурна схема системи управління мобільними агентами де компоненти системи взаємодіють між собою, утворюючи цілісний механізм [10]. Він може бути використаний для оптимізації різноманітних завдань і досягнення

високого рівня ефективності системи управління мобільними агентами. Однак, поточне дослідження не описує рішення з відкритих джерел на базі Kubernetes, що можуть бути використані як основа для впровадження системи, до яких можна віднести легковаговий дистрибутив Kubernetes k3s або систему з відкритим вихідним кодом KubeEdge.

Зараз проведення досліджень щодо розробки нового методу із застосуванням інтегрованих МН і ШІ в керування тактикою бою БпЛА, стійкого до радіоелектронних перешкод є актуальним завданням.

Зв'язок роботи з науковими програмами, планами, темами.

Дослідження, результати яких представлено в дисертаційній роботі, відповідають пріоритетному напряму розвитку науки і техніки України «Інформаційні та комунікаційні технології» і виконувалися відповідно до програм і планів науково-дослідних робіт Державного університету інформаційно-комунікаційних технологій, у тому числі в рамках госпрозрахункової дослідної роботи "Бізнес-аналіз та обґрунтування ІТ рішень для оптимізації виробничих процесів підприємства", в якій автор брав участь як виконавець.

Мета і задачі дослідження. Метою дисертаційної роботи є розробка моделей і методів покращення процесів розробки інформаційних систем МН з використанням сучасних MLOps рішень, таких як Kubeflow, для мобільних агентів. Для досягнення поставленої в роботі мети потрібно вирішити наступні завдання:

1. Проаналізувати сучасний стан розвитку МН і ШІ з можливістю їх застосування в БпЛА.

2. Дослідити можливості використання сучасних MLOps рішень для покращення процесів розробки інформаційних систем МН в розрізі інструментів Kubeflow.

3. Розробити модель побудови та застосування МН із можливістю постійного донавчання, що досягається високим рівнем інтеграції та

автоматизації конвеєру із використанням компонентів Kubeflow і платформи Kubernetes.

4. Експериментально перевірити результати експлуатації моделі в арсеналі Kubeflow із використанням таких факторів покращення як швидкість розробки, імплементації змін, зменшення часу на пошук проблем, відновлення після глобальних перебоїв, зменшення кількості помилок в моделі.

5. Розробити архітектурну концепцію системи на базі дистрибутива Kubernetes k3s з використанням моделі машинного навчання в парадигмі граничних обчислень (Edge computing) для мобільних агентів із використанням mesh-мереж.

6. Провести оцінку ефективності використання бездротової Mesh-мережі для підвищення функціональної стійкості розподіленої інформаційної системи БПлА.

Об'єктом досліджень є процес розгортання конвеєра розробки та використання моделей МН.

Предметом досліджень є методи та алгоритми розробки конвеєра на основі платформи Kubernetes та інструментів Kubeflow із специфікою впровадження для мобільних агентів.

Методи досліджень. Проведені дослідження базуються на методах прийняття рішень, методах штучного інтелекту, комп'ютерного та об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Вперше сформовано концепцію повноцінного інформаційного рішення на базі конвеєру неперервної інтеграції із можливістю постійного донавчання, високого рівня інтеграції та автоматизації з використанням компонентів Kubeflow і платформи Kubernetes, що дозволяє покращити кількісну та якісну складові експериментів, зменшити час, затрачений на його підготовку, та мінімізувати помилки, спричинені людським фактором.

2. Вперше спроектовано архітектурну концепцію системи на базі дистрибутива Kubernetes k3s, що дозволяє ефективно використовувати моделі

МН в парадигмі граничних обчислень (Edge Computing) із використанням Mesh-мереж для мобільних агентів (БпЛА), що формують собою функціонально стійку розподілену інформаційну систему.

3. Удосконалено модель підвищення функціональної стійкості розподіленої інформаційної системи БПлА з використанням бездротової Mesh-мережі передачі даних, що дозволить протистояти радіоелектронним та іншим перешкодам і працювати в автономному або напів-автономному режимі.

Практичне значення одержаних результатів.

Практичне значення одержаних результатів у галузі розробки та створення ефективних методів побудови та використання моделей МН для БпЛА полягає в наступному:

1. Запропоновані архітектурні рішення інформаційних систем та метод можуть бути використані дослідно-конструкторськими організаціями та державними структурами Збройних сил України для реалізації концепції «Аналітична система підтримки прийняття рішень для БпЛА», що показано через функціональні моделі процесів (AS, TO-BE, SHOULD-BE).

2. Сучасні MLOps рішень на базі інструментів Kubeflow для створення та використання навчальних моделей МН впроваджено в дисципліну «Системи підтримки прийняття рішень» в навчальний процес студентів галузі знань 12 «Інформаційні технології» Державного університету інформаційно-комунікаційних технологій. Високий рівень автоматизації та інтеграції компонентів дозволяє створити та запустити тестову модель за відносно короткий час, але без зниження якості її роботи та надійності.

Особистий внесок здобувача. Дисертація є самостійно виконаною завершеною роботою здобувача. Наукові результати і практичні розробки, що містяться в дисертаційній роботі, отримані автором самостійно.

У роботах, опублікованих у співавторстві, автором: [1] – запропоновано схематичний інтегрований конвеєр для розробки та функціонування системи машинного навчання, а також здійснено практичне використання компонента Kubeflow Seldon Core Serving для розгортання тестової моделі; [2] –

запропоновано концепцію використання MLOps конвеєру для розв'язання прикладної задачі класифікації об'єктів із відео розвідувальних БпЛА; [3] – проведено аналіз імплементації та автоматизації розгортання WebRTC (Web Real-Time Communications) SFU (Selective Forwarding Unit) шлюзу в Cloud Native оточенні з акцентом на пошуку оптимального способу розгортання; наведено приклад декларативного опису розгортання SFU шлюзу Jitsi у вигляді коду, що може бути перевикористаний для наступних експериментів; [4] – спроектовано розподілену інформаційну систему на основі мобільних агентів, якими виступають БпЛА, коротко описано їх потенційну практичну реалізацію в умовах реального часу, запропоновано та обґрунтовано використання Mesh-мереж для підвищення функціональної стійкості розподіленої інформаційно системи мобільних агентів; [5] – виконано дослідження сервера WebRTC SFU в кластерному обчислювальному середовищі за допомогою шлюзу Janus WebRTC з використанням постачальника мережі Kubernetes; [7] – доведено, що використання Mesh-мереж дозволяє досягти високої функціональної стійкості інформаційної системи БпЛА з урахуванням затримки між вузлами; [8–11] – запропоновано використання концепції МН і процесів для покращення розробки інформаційних систем різної конфігурації. З робіт, опублікованих у співавторстві, для вирішення задач, поставлених у дисертаційному дослідженні, використано результати, отримані здобувачем особисто.

Апробація роботи. Основні теоретичні положення та практичні результати дисертаційної роботи доповідалися і обговорювались на:

Міжнародній науково-технічній конференції «Перспективи телекомунікацій», Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського (Київ, 2017);

Науково-технічній конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» (Київ, 2020);

Науково-практичній конференції «Проблеми комп'ютерної інженерії» (Київ, 2020);

II Міжнародній науково-практичній конференції «Системний аналіз та інтелектуальні системи для бізнесу та управління» (Київ, 2023).

Публікації. Результати дослідження опубліковані в 11 наукових роботах, з них: одна наукова стаття, що входить до наукометричної бази даних Scopus; три статті у наукових фахових виданнях України; сім тез і матеріалів доповідей конференцій.

Структура та обсяг роботи. Повний обсяг дисертації становить 191 сторінок, з яких 172 сторінок основного тексту. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел за розділами дисертації (111 найменувань) і додатків. Робота містить 53 рисунки, 2 таблиці та 3 додатки.

Список використаних джерел

1. M. Aykol, P. Herring, A. Anapolsky, “Machine learning for continuous innovation in battery technologies”. Nat. Rev. Mater, vol. 5, 2020, no. 10, pp. 725–727.

2. M. K. Gourisaria, R. Agrawal, G. M. Harshvardhan, M. Pandey, S. S. Rautaray, “Application of Machine Learning in Industry 4.0. Machine Learning: Theoretical Foundations and Practical Applications”. Springer, 2021, pp. 57–87.

3. A. De Las Heras, A. Luque-Sendra and F. Zamora-Polo, “Machine learning technologies for sustainability in smart cities in the post-covid era”. Sustainability. 2020, no. 12, 22, pp. 9320.

4. Google Cloud. MLOps: Continuous Delivery and Automation Pipelines in Machine Learning. [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (дата звернення 01.11.2023).

5. D. Kreuzberger, N. Kühl, S. Hirschl, “Machine Learning Operations (MLOps): Overview, Definition, and Architecture”. Computer Science: Machine Learning, vol. 11, 2023, pp. 31866.

6. D. Sculley at all, “Hidden Technical Debt in Machine Learning Systems. Part of Advances in Neural Information Processing Systems”, NIPS. 2015, 1486, 9 p.

7. A. I. Khan, Y. Al-Mulla, “Unmanned Aerial Vehicle in the Machine Learning Environment”. Procedia Computer Science, vol. 160, 2019, pp. 46–53.

8. M. Samuel S., C. M. Sibley, J. T. Coyne, “Using machine learning and real-time workload assessment in a high-fidelity UAV simulation environment”. Proceedings SPIE: Next-Generation Analyst IV, vol. 9851, 2016, 98510B.

9. J. Moeyersons, B. Stamper, B. Volckaert, F. De Turck, “Implementing a Network-Aware Kubernetes Scheduler on top of a Mesh Network”, 2022 IEEE Cloud Summit.

10. Яровий О. В., Завгородній В. В., “Розробка системи управління мобільними агентами”. Таврійський науковий вісник. Серія: Технічні науки, 2023, № 3, с. 29–36.

РОЗДІЛ 1
АНАЛІЗ РОЗВИТКУ ОСНОВНИХ ТИПІВ МАШИННОГО НАВЧАННЯ

1.1 Розвиток і сфери застосування машинного навчання

Машинним навчанням є отриманням знань із даних. Цей термін вперше був введений у кінці 1950-х років А. Семюелем. МН є узагальненим терміном стосовно розв’язування задач, для яких розробка алгоритмів програмістами була б занадто витратною. МН є дослідженнями на стику статистики, штучного інтелекту та інформатики і також відома як прогнозована аналітика або статистичне навчання. Застосування методів машинного навчання набирає своєї популярності. Багато сучасних веб-сайтів і пристроїв мають у своїй основі алгоритми машинного навчання: від автоматичних рекомендацій, які фільми дивитися, яку їжу замовляти чи які продукти купувати до персоналізованого онлайн-радіо й розпізнавання друзів на фотографіях тощо [1–3].

МН відоме у своєму застосуванні в комерційних задачах під назвою «передбачувальна аналітика» (predictive analytics). Не все МН ґрунтується на статистиці, а обчислювальна статистика (statistical computing) є важливим джерелом методів цієї галузі. Математичні основи МН забезпечують методи математичної оптимізації. Добуванням даних (data mining) є дослідженням, які зосереджене на розвідувальному аналізі даних через некероване навчання [4].

Окрім комерційних застосувань МН мало величезний вплив на сьогоднішній спосіб проведення досліджень на основі даних. Системи МП були застосовані до різноманітних наукових проблем, таких як розуміння зірок, пошук далеких планет, відкриття нових частинок, аналіз послідовностей ДНК і надання індивідуальних методів лікування раку.

Якщо розглянути будь який складний веб-сайт, наприклад, такий як Facebook, Amazon або Netflix, дуже ймовірно, що кожна частина сайту містить кілька моделей МН. Алгоритми МН, наприклад, можна використовувати для виявлення шахрайських фінансових операцій [5], прогнозувати відтік клієнтів,

оптимізувати управління ланцюгом поставок [6], та автоматизувати виробничі процеси [7].

В останній половині XX століття більшість робочої сили в розвиненому світі перейшла від фізичної праці до так званої роботи знань. Чіткі визначення «перенести це звідси туди» та «проробити в цьому дірку» зникли. Зараз усе набагато неоднозначніше. Робочі завдання, такі як «максимізація прибутку», «мінімізація ризику» та «знайти найкращу маркетингову стратегію», дуже поширені. Безкінечний потік інформації, доступної нам із мережі, робить роботу працівників інтелектуальної сфери ще важчою.

На початку «інтелектуальних» програм багато систем використовували закодовані вручну правила рішень «if» і «else» для обробки даних або пристосування до введення користувача. Прикладом може бути спам-фільтр, завданням якого є переміщення відповідних вхідних повідомлень електронної пошти до папки зі спамом. Можна скласти чорний список слів, які призведуть до того, що електронний лист буде позначено як спам. Це був би приклад використання системи правил, розробленої експертами, для розробки «інтелектуальної» програми.

Розробка правил прийняття рішень вручну можлива для деяких програм, особливо для тих, у яких люди добре розуміють процес моделювання. Однак використання закодованих вручну правил для прийняття рішень має два основні недоліки:

логіка, необхідна для прийняття рішення, є специфічною для окремої області та завдання, а навіть незначна зміна завдання може вимагати перепису всієї системи;

розробка правил вимагає глибокого розуміння того, як експерт повинен приймати рішення.

Одним із прикладів того, де цей ручний підхід не працює, є виявлення обличчя на зображеннях. Майже кожен смартфон може визначити обличчя на зображенні, проте розпізнавання обличчя залишалося невирішеною проблемою аж до 2001 року. Основна проблема полягає в тому, що спосіб «сприйняття»

пікселів (з яких складається зображення в комп'ютері) комп'ютером сильно відрізняється від того, як сприймають люди обличчя. Ця різниця в представленні робить фактично неможливим для людини придумати гарний набір правил для опису того, що являє собою обличчя в цифровому зображенні. Однак, використовуючи МН, достатньо просто представити програму з великою колекцією зображень обличчя, щоб алгоритм визначив, які характеристики необхідні для ідентифікації обличчя.

Самонавчання (self-learning) як парадигма МН було запропоновано у 1982 році разом із нейронною мережею, здатною до самонавчання, яка отримала назву поперечинного адаптивного масиву [8] Це навчання без зовнішніх винагород і без зовнішніх порад вчителя. Алгоритм самонавчання у рамках поперечинного адаптивного масиву обчислює, поперечинним чином, як рішення щодо дій, так і емоції (почуття) щодо наслідкових ситуацій [9].

Найуспішнішими видами алгоритмів МН є ті, які автоматизують процеси прийняття рішень шляхом узагальнення відомих прикладів. У цьому параметрі, який відомий як контрольоване навчання, користувач надає алгоритму пари вхідних даних і бажаних виходів, і алгоритм знаходить спосіб отримати бажаний результат за вхідних даних. Зокрема, алгоритм може створити вихід для вхідних даних, яких він ніколи раніше не бачив, без будь-якої допомоги людини.

Повертаючись до прикладу класифікації спаму, використовуючи МН, користувач надає алгоритму велику кількість електронних листів (які є вхідними) разом із інформацією про те, чи є якісь із цих електронних листів спамом (що є бажаним результатом). Отримавши новий електронний лист, алгоритм створить прогноз щодо того, чи є новий електронний лист спамом.

1.2 Основні типи машинного навчання

МП є великою сферою, яка поділяється на п'ять основних типів: кероване навчання (supervised learning); некероване навчання (unsupervised learning); глибоке навчання (deep learning); навчання з підкріпленням (reinforcement learning); природна мова обробки NLP (Natural Language Processing). Також існує

багато інших типів МН, які можна реалізувати на конкретних наборах даних (рис. 1.1).

Рис. 1.1 Основні типи МН та їх різновиди

Алгоритми керованого навчання базуються на математичній моделі набору даних, що містить як входи, так виходи [10–13]. Тренувальні дані, які складаються з набору тренувальних прикладів, використовуються у такій моделі. Це приклад має один або кілька входів та бажаний вихід, відомий також як керівний сигнал. Математично кожен тренувальний приклад подається чи масивом, чи вектором, який називається вектором ознак, а тренувальні дані

подаються матрицею. Завдяки ітераційній оптимізації цільової функції алгоритми керованого навчання навчаються функції, яку можливо використовувати для передбачування виходу, що пов’язаний з новими входами [14].

У МН «кероване навчання» є навчанням, в якому набір позначених даних (відомих екземплярів) надається моделі для встановлення прийнятних зв’язків між функціями. З цих зв’язків модель може передбачити результати для нових наборів даних на основі зв’язків, розроблених для існуючих наборів даних. Кероване навчання є потужним інструментом для класифікації та обробки даних. З іншого боку, неконтрольоване навчання є типом МН, який використовується для встановлення зв’язків із наборів даних, що складаються з екземплярів без позначених відповідей. Це можна назвати навчанням машини з використанням інформації, яка не є ані секретною, ані маркованою, і дозволяє бажаному алгоритму створювати зв’язки без будь-яких попередніх знань.

Кероване навчання розгалужується ще на два типи: регресори та класифікатори, які є найпоширенішим і ефективним способом прогнозування результатів на основі наданого набору даних (позначених даних) у машинному навчанні. Регресори та класифікатори мають додаткові підтипи, які мають різні алгоритми та методи встановлення зв’язків у наборах даних.

Основне завдання некерованого навчання полягає в тому, щоб організувати інформацію в групи відповідно до подібності, закономірностей і зв’язків між ознаками без будь-якого керівництва (програми). МН в основному включає п’ять процесів для отримання результатів.

1. Перший крок – це *збір даних*. Надання автентичних даних є необхідною умовою для застосування моделі МН. Залежно від бажаного алгоритму МН потрібні типи наборів даних. У контрольованому навчанні потрібні позначені набори даних, а у випадку неконтрольованого навчання немає необхідності надавати набори даних із позначеними відповідями.

2. Другий крок – важливим крок у МН, тобто є етапом *попередньої обробки даних*. На цьому кроці наданий набір даних змінюється відповідно до вимог. Цей

крок включає заміну відсутніх значень бажаною технікою (зазвичай береться середнє), обробку категоріальних даних, масштабування даних за функціями (важливий крок для деяких моделей), поділ набору даних на набір для навчання та набір для тестування тощо.

3. Третій крок – це *вибір відповідної техніки МН* залежно від потреби та типу бажаного результату.

4. Четвертим кроком є *застосування алгоритму МН* до набору даних (зазвичай застосовується до навчального набору) для моделі, щоб створити необхідні зв'язки та шаблони.

5. Після того, як модель дізнається про співвідношення для наданих наборів даних, нові дані можуть бути надані в модель для *прогнозування результату*.

Навчання з підкріпленням – це галузь МН, пов'язана виконанням дії в середовищі програмними агентами з метою максимізації певного уявлення про сукупну винагороду. Завдяки універсальності цей тип МН вивчають у багатьох інших дисциплінах, таких як теорія керування, дослідження операцій, теорія ігор, теорія інформації, оптимізація на основі моделювання, статистика та генетичні алгоритми тощо [15, 16] Алгоритми навчання з підкріпленням використовують тоді, коли точні моделі недосяжні. Наприклад, їх використовують в автономних транспортних засобах і при навчанні гри проти людини-супротивника.

Залежно від попередньої обробки даних і вибору алгоритмів МН досягається точність, точність та інші показники. На рис. 1.2 показано порядок кроків у моделі МН: попередня обробка; структурування даних; алгоритм МР; модель-кандидат; розгортання обраної моделі.

Рис. 1.2 Порядок кроків у моделі МН

1.3 Різновиди машинного навчання

1.3.1 Кероване і некероване машинне навчання

Алгоритми МН, які навчаються на парах введення/виведення, називаються алгоритмами кероване навчання, оскільки «вчитель» забезпечує нагляд за алгоритмами у формі бажаних результатів для кожного прикладу, з якого вони навчаються. Хоча створення набору даних вхідних і вихідних даних часто є трудомістким ручним процесом, алгоритми навчання під наглядом добре зрозумілі, а їх продуктивність легко виміряти. Якщо програма може бути сформульована як керована навчальна проблема, і можна створити набір даних, який містить бажаний результат. МН, ймовірно, зможе вирішити певну проблему [10–13].

Приклади керованих завдань МН включають:

1. *Визначення поштового індексу за рукописними цифрами на конверті*. Тут введенням є сканований почерк, а бажаним виходом є фактичні цифри поштового індексу. Щоб створити набір даних для побудови МН, потрібно зібрати багато конвертів. Тоді ви зможете самостійно прочитати поштові індекси та зберегти цифри як бажані результати.

2. *Визначення доброякісності пухлини на основі медичного знімка*. Тут вхідними даними є зображення, а вихідними – чи є пухлина доброякісною. Щоб створити набір даних для побудови моделі, потрібна база даних медичних зображень. Вам також потрібен експертний висновок, тому лікар повинен переглянути всі зображення та вирішити, які пухлини є доброякісними, а які ні. Може навіть знадобитися провести додаткову діагностику, крім вмісту зображення, щоб визначити, чи є пухлина на зображенні раковою чи ні.

3. *Виявлення шахрайської діяльності в транзакціях з кредитними картками*. Тут вхідними даними є запис транзакції кредитної картки, а вихідними – чи є ймовірність шахрайства. Якщо припустити, що ви є організацією, яка розповсюджує кредитні картки, збір набору даних означає збереження всіх транзакцій і запис, якщо користувач повідомляє про будь-яку транзакцію як шахрайську.

У цих прикладах варто зауважити цікаву річ: хоча вхідні та вихідні дані виглядають досить зрозуміло, процес збору даних для цих трьох завдань значно відрізняється. Хоча читання конвертів трудомістке, це легко та дешево. З іншого боку, отримання медичних зображень і діагностики вимагає не тільки дорогого обладнання, але й рідкісних і дорогих експертних знань, не кажучи вже про етичні проблеми та проблеми конфіденційності. У прикладі виявлення шахрайства з кредитною карткою збір даних набагато простіший. Клієнти надають бажаний результат, оскільки вони повідомлять про шахрайство. Усе, що вам потрібно зробити, щоб отримати пари введення/виведення шахрайської та нешахрайської діяльності, це почекати.

Некеровані алгоритми є ще одним з відомих типів алгоритмів. У некерованому навчанні відомі лише вхідні дані, а відомі вихідні дані не надаються алгоритму [10–13]. Хоча є багато успішних застосувань цих методів, зазвичай їх важче зрозуміти й оцінити.

Приклади навчання без контролю такі:

1. *Визначення тем у наборі публікацій блогу*. Якщо у вас є велика колекція текстових даних, ви можете узагальнити її та знайти в ній поширені теми. Можливо, ви не знаєте наперед, що це за теми або скільки їх може бути. Тому немає відомих результатів.

2. *Сегментація клієнтів на групи зі схожими перевагами*. Маючи набір записів про клієнтів, ви можете визначити, які клієнти схожі, і чи існують групи клієнтів із подібними перевагами. Для торгового сайту це можуть бути «батьки», «книжкові хробаки» або «геймери». Оскільки ви не знаєте наперед, якими можуть бути ці групи або навіть скільки їх, у вас немає відомих результатів.

3. *Виявлення ненормальних моделей доступу до веб-сайту*. Щоб виявити зловживання або помилки, часто корисно знайти моделі доступу, які відрізняються від норми. Кожен ненормальний шаблон може дуже відрізнятися, і ви можете не мати жодних зареєстрованих випадків аномальної поведінки. Оскільки в цьому прикладі ви лише спостерігаєте за дорожнім рухом і не знаєте, що є нормальною та ненормальною поведінкою, це неконтрольована проблема.

Як для керованих, так і для некерованих навчальних завдань важливо мати представлення ваших вхідних даних, яке може зрозуміти комп'ютер. Часто буває корисно розглядати дані як таблицю. Кожна точка даних, про яку йдеться (кожна електронна пошта, кожен клієнт, кожна транзакція), є рядком, а кожна властивість, яка описує цю точку даних (скажімо, вік клієнта або сума чи місце транзакції), є колонка. Можна описувати користувачів за їхнім віком, статтю, коли вони створили обліковий запис і як часто вони купували у онлайн-магазині. Можете описати зображення пухлини за допомогою значень градацій сірого кожного пікселя або, можливо, за допомогою розміру, форми та кольору пухлини. Кожна сутність або рядок відомі як зразок (або точка даних) у МН, тоді як стовпці – властивості, які описують певні сутності – функції.

Побудова гарного представлення даних полягає у вилученні ознак або розробці функцій. Однак, жоден алгоритм МН не зможе зробити прогноз на основі даних, для яких у нього немає інформації. Наприклад, якщо єдиною характеристикою пацієнта є його прізвище, жоден алгоритм не зможе передбачити його стать. Ця інформація просто не міститься у наявних даних. Якщо додати іншу функцію, яка містить ім'я пацієнта, то можна визначити стать людини за ім'ям.

1.3.2 Глибинне навчання

Глибинне навчання може бути як навчанням із вчителем так і без одночасно. Воно також відоме як глибинне структурне навчання (deep structured learning), ієрархічне навчання (hierarchical learning), глибинне МН (deep machine learning). Це галузь МН, що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень [17–22].

Глибинне навчання є частиною ширшого сімейства методів МН, що ґрунтуються на навчанні ознак даних. Спостереження (наприклад, зображення) може бути представлено багатьма способами, такими як вектор значень

яскравості для пікселів, або абстрактнішим способом, як множина кромок, областей певної форми тощо. Деякі представлення є кращими за інші у спрощенні задачі навчання (наприклад, розпізнаванню обличчя, або виразів обличчя). Однією з обіцянок глибинного навчання є заміна ознак ручної роботи дієвими алгоритмами автоматичного або напівавтоматичного навчання ознак та ієрархічного виділяння ознак.

Дослідження в цій галузі намагаються зробити кращі представлення та створити моделі для навчання цих представлень з великомасштабних немічених даних. Деякі з цих представлень було зроблено під натхненням досягнень в нейронауці та з мотивів схем обробки та передавання інформації в нервовій системі, таких як нервове кодування, що намагається визначити зв’язок між різноманітними стимулами та пов’язаними нейронними реакціями в мозку.

Різні архітектури глибинного навчання, такі як глибинні нейронні мережі, згорткові глибинні нейронні мережі, глибинні мережі переконань та рекурентні нейронні мережі застосовувалися в таких областях, як комп’ютерне бачення, автоматичне розпізнавання мовлення, обробка природної мови, розпізнавання звуків та біоінформатика, де вони, як було показано, представляють передові результати в різноманітних задачах.

Глибоке навчання як напрямок машинного навчання лише нещодавно набув популярності. Якщо говорити точніше, індустрії в усьому світі почали застосовувати методи глибокого навчання лише через десять років у XXI столітті. На відміну від інших методів МН, алгоритми глибокого навчання розроблені таким чином, що людям не потрібно вказувати, які характеристики стосуються конкретного об’єкта для прогнозування. Наприклад, МН не може дуже точно передбачити правильний об’єкт, враховуючи його зображення, лише через величезну кількість (скажімо, тисячі) функцій, які потрібно вивчити, і вони також можуть відрізнятися від об’єкта до об’єкта. Щоб відрізнити котів від собак, модель має вивчити риси обличчя, але не зовсім так у класифікації автомобілів від мотоциклів.

Таким чином, глибоке навчання використовує нейронні мережі для автоматизації процесу виявлення функцій і дозволяє комп'ютеру вирішувати, які функції важливі, а які ні. Воно має багато практичних застосувань в Індустрії 4.0 і відіграє життєво важливу роль у медичних галузях для отримання точних результатів. Нейронні мережі моделюються за моделлю людського мозку, щоб спробувати імітувати взаємозв'язок різних нейронів, які спрацьовують на основі вхідних даних, які вони отримують. Штучна нейронна мережа (ШНМ, Artificial Neural Network) вважається основою глибокого навчання. ШНМ також називають багатошаровим перцептроном, який винайшов ще Ф.Ф. Розенблат ще в 1958 році [23, 24].

Згорточна нейронна мережа CNN (Convolutional Neural Network,) є типом архітектури нейронної мережі глибокого навчання, який зазвичай використовується в комп'ютерному зорі [25]. Ця мережа використовує різновид багатошарових перцептронів, розроблений так, щоб використовувати мінімальний обсяг попередньої обробки. Комп'ютерний зір є сферою штучного інтелекту, яка дозволяє комп'ютеру розуміти та інтерпретувати зображення або візуальні дані. CNN є одним із найпотужніших інструментів МН з точки зору регресивних і класифікаційних можливостей. Вона складається з ряду взаємопов'язаних вузлів, які організовані пошарово, як показано на рис. 1.3.

З'єднання між вузлами визначаються як ваги, які коригуються кожного разу, коли вхідні дані передаються туди-сюди через мережу. Вхідні дані подаються в мережу на вхідному рівні (самий лівий рівень) у масштабованій формі, яка проходить через вузли та множиться на ваги між з'єднаннями двох вузлів у послідовних шарах. У математичних термінах нехай L позначає набір шарів CNN, заданий L = {L1, L2,..., LM}, де M – загальна кількість шарів.

За замовчуванням ми класифікуємо типи шарів на три: вхідний, прихований і вихідний. Вихідний рівень визначається як крайній правий рівень, який забезпечує softmax (означає те, що всі індивідуальні значення, коли підсумовуються, дорівнюватимуть одиниці) ймовірності кожного класу. Кількість вузлів у вихідному рівні зазвичай дорівнює числу класів, так що кожен

вузол представляє клас. Найвище значення серед усіх вузлів, узятих на вихідному рівні, забезпечує прогнозований клас. Усі шари, що знаходяться між вихідним і вхідним шарами, називаються прихованими шарами, які не можна спостерігати.

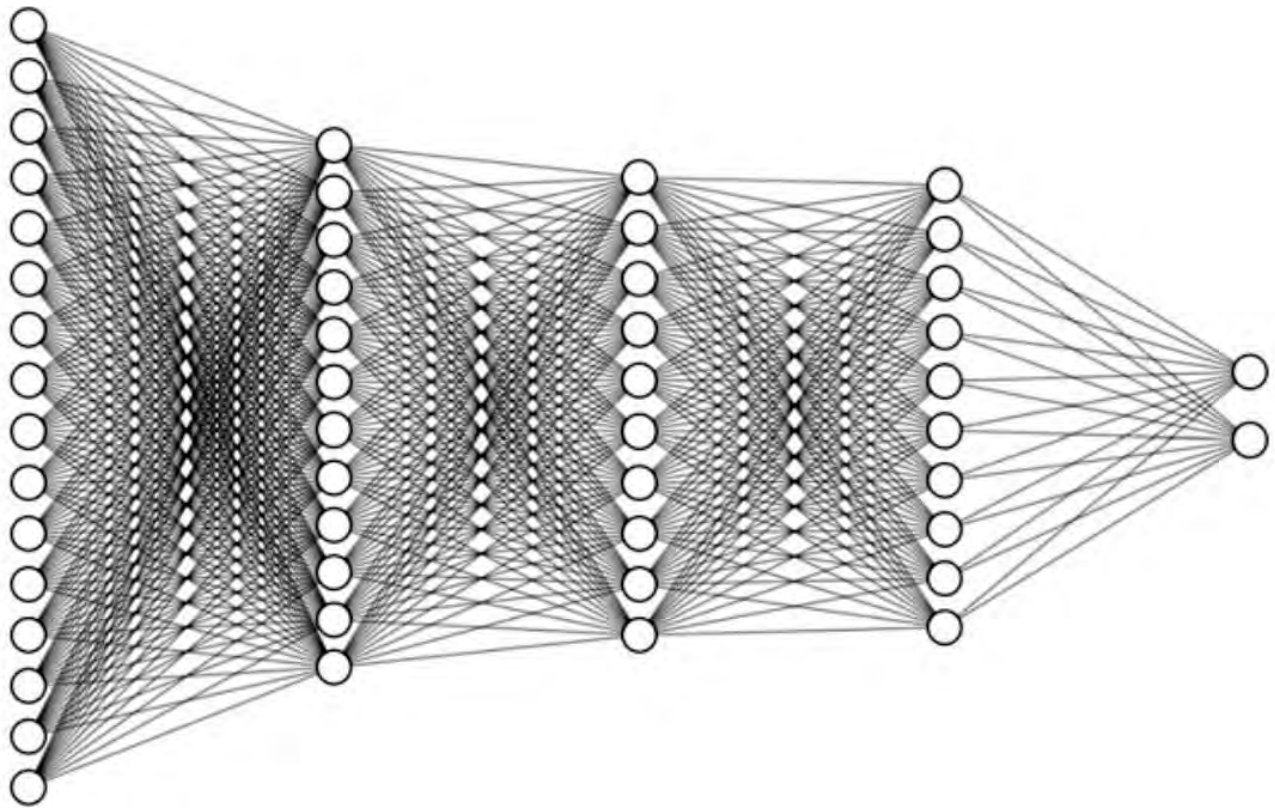


Рис. 1.3 Вузли згорточної нейронної мережі

1.4 Концепція спільного підходу до процесів машинного навчання та експлуатації

1.4.1 Розвиток моделей машинного навчання та експлуатації

Наука про дані (Data Science), аналітика даних (Data Analytics) і МН на сучасному етапі розвитку інформаційних технологій стають одними із головних інструментів для розв’язання складних прикладних задач у різних сферах діяльності. МН є одним із методів функціонування штучного інтелекту, а саме – практичної реалізації його можливостей шляхом створення алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальше використання для самонавчання [26–28]. Загалом, обидві галузі пов’язані з

використанням обчислювальних і математичних підходів для покращення прийняття рішень.

Спостерігається швидкий прогрес у різних галузях досліджень МН таких як комп'ютерний зір, розуміння природної мови та рекомендаційні системи штучного інтелекту. Тому багато компаній інвестують у свої команди з обробки даних і можливості МН, щоб розробити прогнозні моделі, які можуть принести бізнес-цінність своїм користувачам.

Для розробки, тестування та підтримки інфраструктури системи із великими даними застосовуються різні процеси та технології.

Запорукою ефективності МН стали такі чинники, як:

- доступ до великих наборів даних;
- невисока ціна обчислювальних ресурсів;
- готові сервіси на основі хмарних технологій;
- швидкий прогрес у сферах штучного інтелекту, такі як комп'ютерний зір, розуміння природної мови та рекомендаційні системи.

Разом із тим у розробці МН розробки існує велика кількість додаткових складнощів, через це безліч МН систем зазнає невдачі на ранніх стадіях розробки і не досягає продуктового середовища.

Для вирішення цих проблем використовують концепцію спільного підходу до процесів МН і експлуатації MLOps, яка є набором технік для імплементації та автоматичної неперервної інтеграції, а також доставки на продуктове середовище та навчання моделей. MLOps є інженерною культурою та практикою МН, спрямованою на уніфікацію розробки МН системи (Dev) і їх експлуатацію (Ops). Практика MLOps означає, що необхідно застосовувати автоматизацію і моніторинг на всіх етапах створення системи МН, включаючи інтеграцію, тестування, випуск, розгортання та управління інфраструктурою.

Фахівці з обробки даних можуть реалізувати і навчити модель МН із прогнозованою ефективністю на офлайн-затриманому наборі даних, маючи відповідні навчальні дані для свого сценарію використання. Однак справжнє завдання полягає не в побудові моделі МН, а в побудові інтегрованої системи

МН і її безперервній експлуатації у виробництві. Завдяки довгій історії виробничих служб МН компанією Google виявлено, що під час експлуатації систем на основі МН у виробництві може бути багато підводних каменів. Деякі з цих «підводних каменів» узагальнено в «Машинне навчання: кредитна картка з високими відсотками технічного боргу» [29].

Розробники програмного забезпечення все більше віддають перевагу концепції контейнеризації. Компанія Google є однією з небагатьох, якій необхідно керувати розгортанням і розробкою великої кількості компонентів сервісу на сотнях тисяч серверів. В основі концепції контейнеризації та мікросервісів була розроблена розподілена система керування контейнерами з відкритим кодом під назвою Kubernetes. Останній може не лише підтримувати повну незалежність програм, але й покращувати використання апаратних ресурсів, тому його широко використовують більшість установ.

Таким чином, концепцію MLOps доцільно розглядати в розрізі інструментів Kubeflow – Cloud Native системи з відкритим кодом, що працює на платформі Kubernetes. Проект Kubeflow присвячений спрощенню розгортання робочого циклу МН систем в Kubernetes. Завдання останнього не дублювати інші служби, а забезпечити простий спосіб розгортання найкращих систем з відкритим кодом для МН на різних інфраструктурах, будь то приватні сервери на виробництві або публічні хмарні рішення. Тобто там, де працює Kubernetes, може працювати Kubeflow, що робить його гнучким та портативним.

1.4.2 Аналіз сучасних досліджень у галузі розробки та експлуатації моделей машинного навчання

МН довело свою ефективність в сучасних реаліях, а інвестиції в розробку якісних, передбачуваних моделей виросли в рази. Процес розробки моделі по своїй суті відповідає конвеєру на виробництві, як ланка послідовних або паралельних етапів [26–28]. Концепція MLOps є набором технік для імплементації та автоматичної неперервної інтеграції, доставки та навчання моделей, яка охоплює основні патерни та антипатерни, на основі яких будують

успішний конвеєр автоматизації розробки систем МН. Ефективність використання підходу MLOps було розглянуто і доведено в багатьох роботах, де було формалізовано основні вимоги до систем МН [29, 30].

Проте разом із новими можливостями приходять додаткові складнощі. Ця проблема поряду із іншими, була розкрита у роботі, написаній працівниками Google щодо невидимого технічного боргу в МН системах (рис. 1.4) [31].

Рис. 1.4 Сучасна МН система

Слід зазначити, що хоча термін MLOps не фігурує в згаданій статті, але основні патерни та антипатерни, на основі яких будуються успішний конвеєр актуальні й зараз. Сучасні МН системи складаються із великої кількості компонентів. МН код є тільки невеликою частиною цієї системи і тому зображений як маленький чорний чотирикутник в центрі (рис. 1.4). Для повноцінної роботи МН систем потрібна складна і велика інфраструктура.

Опис концепцій МН для промислової інженерії та дослідження операцій з використанням реальних програм у вигляді курсів для студентів розглянуто у [31]. Такі дослідження підвищують інтерес студентів до МН і оцінки реального впливу, який може мати аналітика. Це сприяє розвитку практичних навичок, а у поєднанні МН і дослідження операцій може бути цінним доповненням до навчальних програм для студентів. Однак, наразі обмежена наявність освітньої дослідницької літератури про навчання щодо МН, яку можна використати для

розробки курсів, особливо для спеціальностей, не пов'язаних з комп'ютерними науками. Цим проблемам приділено увагу у [32, 33].

В [34] проведені дослідження того як студенти формують концепцію свого розуміння методу МН без нагляду після взаємодії з інтерактивними візуалізаціями, розробленими з використанням підходу обчислювального когнітивного учнівства, для інтеграції знань концепцій Data Science. А також досліджено як можливості інтерактивної візуалізації підтримують або перешкоджають інтеграції знань студентів у некерованому навчанні. Однак дане дослідження не робить прив'язки до сучасних МН систем розробки, що не дозволяє нам оцінити його із практичної точки зору. Було б доцільно оцінити одну із поточних МН систем (Kubeflow, MLFlow) з точки зору легкості сприйняття для учнів або студентів.

Завдяки зростаючій доступності даних і досягненнями у МН і методології оптимізації аналітика даних ширше використовується для вирішення проблем управління операціями. Поширений підхід до вирішення реальних проблем операційного менеджменту за алгоритмом «передбачити, потім оптимізувати» розглянуто в [35, 36]. У такому підході ключові параметри оптимізаційної моделі прогнозуються за допомогою МН до або одночасно з вирішенням оптимізаційних моделей. В [35, 36] розглянуто застосування аналітики даних для управління операціями в трьох основних сферах – управління ланцюгом поставок, управління доходами та операції з охорони здоров'я. Однак у цих працях не розглядаються будь-які конкретні рекомендації щодо покращення або зміни моделі, наприклад, як реакцію на зміну ланцюгів поставок.

В [37] розглянуті основні концепції, спільні для багатьох алгоритмів МН, зокрема налаштування гіперпараметрів, як самі по собі є важливими цілями навчання, незалежно від алгоритмів МН. Обговорено типи МН, їхні параметри та гіперпараметри, навчання моделі, валідація та тестування. При цьому автори наголошують, що незважаючи на те, що постійно докладаються зусилля, щоб покращити МН алгоритми та зробити їх більш зрозумілими та інтерпретованими, роль викладачів полягає в покращенні розуміння людиною МН алгоритмів.

В [38] запропонована структура конвеєра оновлення моделі та керування для моделей AIOps (Artificial Intelligence and Operation) у системах мікросервісів. Це зроблено з метою навчання, інкапсулювання та розгортання моделі, а також для спрощення процесу. Для забезпечення попередньої реалізації фреймворку та перевірки розроблений прототип системи на основі Kubernetes і Gitlab. Процес пакування та розгортання автоматизовано за допомогою технології безперервної інтеграції. Ця робота є корисним ресурсом для побудови інтегрованої та оптимізованої системи керування моделлю AIOps.

Однак, у запропонованій технології присутній недостатній рівень інтегрованості між ланками та низький рівень використання абстракцій Kubeflow для спрощення моделі. Наприклад, замість написання Dockerfile для моделі можна використати компонент Seldon Core. Також у роботі не було розкрито концепцію автоматичного донавчання моделі. Адже Kubeflow конвеєр може запускатись по тригеру, яким виступає наявність нових даних для навчання. Ці зміни додали б описаній МН системі більше універсальності та портативності, оскільки вона автоматично адаптовувалась під конкретне середовище мікресовервісів і не потребувала ручного донавчання.

Компонент конвеєра, включений у Kubeflow, може створити цілий робочий процес МН. Однак цей компонент усе ще залежить від Google Cloud Platform, тому недостатньо дружній для розробників, які не мають умов для його оренди. В [39] досліджені проблеми конвеєра Kubeflow, перевірено здійсненність рішення незалежності Kubernetes від Google Cloud Service на прикладі підтримки програми МН на основі Pytorch. Це рішення дає можливість науковцям з обробки даних створювати програми глибокого навчання на основі Pytorch у розробці розподілених систем на основі Kubernetes. Запропонований підхід забезпечує стабільну та безперервну роботу програми та розумне планування на кластері, що містить різнорідні обчислювальні ресурси.

В [40] описано можливість використання Mesh-мереж для платформи Kubernetes та потенційні переваги використання оновленого компонента планувальника навантаження із можливістю оцінювати затримки між вузлами.

Запропоноване рішення може покращити загальну продуктивність розподіленої системи коли буде виникати необхідність у збільшенні кількості вузлів. Однак в роботі не в достатній мірі розглянуто практичні переваги застосування Mesh-мереж в Kubernetes та дистрибутиви, розроблені для систем в парадигмі кордонних обчислень (Edge computing).

В [41] описано структурна схема системи управління мобільними агентами. Компоненти системи взаємодіють між собою, утворюючи цілісний механізм, який може бути використаний для оптимізації різноманітних завдань та досягнення високого рівня ефективності системи управління мобільними агентами. Однак, поточне дослідження не описує рішення з відкритих джерел на базі Kubernetes, що можуть бути використані як основа для впровадження системи, до яких можна віднести легковаговий дистрибутив Kubernetes k3s або систему з відкритим вихідним кодом KubeEdge.

В [42] запропонована структура життєвого циклу МН MLife (Mashine Learning Life) – циклічного процесу створення ефективної системи МН. Структура базується на тому факті, що потік даних у MLife є замкнутим циклом, керованим невдалими випадками. Вони найбільше впливають на продуктивність моделі МН, однак також забезпечують найбільшу цінність для подальшої розробки моделі МН. Це дозволяє підприємствам швидко відстежувати їхні можливості МН. Однак, запропонована структура MLife підходить лише для систем МН на їх ранніх стадіях.

1.5 Автоматизація розгортання на прикладі WebRTC застосунків

Останніми роками прослідковується значний інтерес до технологій передачі потокових даних, викликаний вимогами часу. Потреба в ефективному, надійному та безпечному каналі спілкування існує у різних сферах життєдіяльності, зокрема у державних структурах, у військових та приватних компаній. Віддалена робота також вносить корективи в різні сфери діяльності. Наприклад, компанія Twitter дозволила віддалено працювати своїм працівникам

незалежно від карантинних обмежень, як і багато інших роботодавців, оскільки вбачають в цьому позитивні моменти [43].

Про зростання інтересу до даних технологій свідчить зростання цін акцій компаній на фондовому ринку, що спеціалізується на відеоконференціях. Тому актуальність застосування технологій передачі потокових даних наразі доволі висока, а необхідність у швидкому та ефективному розгортанні є на часі, що може знайти своє відображення у використанні Cloud Native екосистеми (хмарних середовищ).

Застосування та розвиток систем на основі протоколу WebRTC для аудіо- та відео конференцій на сьогодні є актуальним і затребуваним. Протокол технології Web комунікації в реальному часі WebRTC дозволяє веб-браузерам взаємодіяти через прикладні програмні інтерфейси API (Application Programming Interface), що в основному використовуються для аудіо- та відеоконференцій. Шлюз вибіркового направлення SFU і шлюз багатоточкового контролю MCU (Multipoint Control Unit) є технологіями, які доповнюють WebRTC.

На рис. 1.5, а видно, що при зростанні кількості користувачів в рамках однієї сесії, кількість одночасних з’єднань зростає в геометричній прогресії від кількості клієнтів, збільшуючи тим самим навантаження на мережу та систему в цілому. Як показано рис. 1.5, б) використання SFU, дозволяє зменшити кількість одночасних сесій. Отже, SFU і MCU сервери використовуються для покращення роботи при впливі неоднорідностей оточення, таких як збільшення кількості користувачів, специфічне мережеве обладнання, але, через складність їх роботи, розгортання в Cloud Native оточенні залишається нетривіальною задачею [44–46].

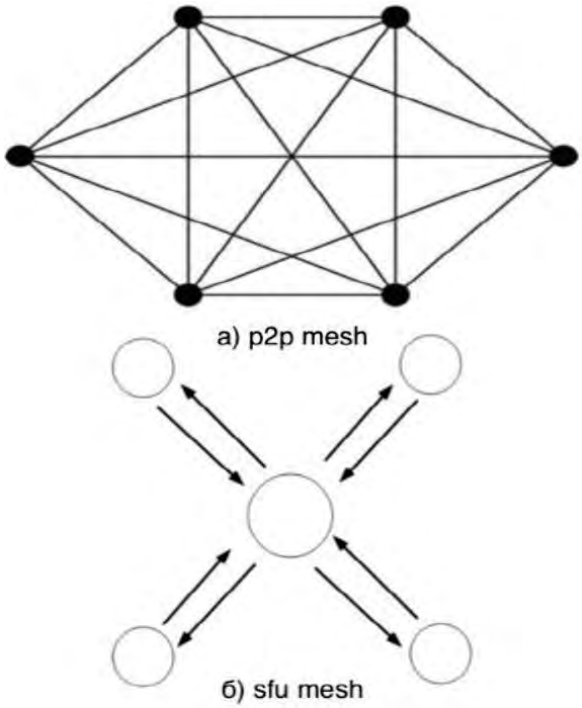


Рис. 1.5 Схематичне відображення з’єднань p2p mesh (один з одним) (а) та із використанням технології SFU (б)

Протокол WebRTC дозволяє веб-браузерам взаємодіяти в реальному часі через прикладні програмні інтерфейси API – інтерфейси JavaScript. Вони є однією із ключових в області аудіо- та відеоконференцій, але мають певні недоліки. Зокрема, при зростанні кількості користувачів в рамках однієї сесії, кількість одночасних з’єднань зростає в геометричній прогресії від кількості клієнтів і тим самим значно збільшуючи навантаження на мережу та систему в цілому. Цю проблему було вирішено у роботі [47] з використанням SFU і MCU серверів, які в свою чергу є складними інфраструктурними компонентами.

В [48, 49] досліджено переваги та специфіка протоколу WebRTC, його можливості для бізнесу та медицини з використанням WebRTC-сумісного веб-браузера Google Chrome або Firefox. В [50] розглянуті питання вирішення проблем багатоточковості аудіо- та відеоконференцій з використанням технології SFU і MCU. Запропоновано підхід при якому модуль MCU інтегрований у браузер для змішування та транскодування відео та аудіо потоків у режимі реального часу.

Сам процес розгортання SFU і MCU застосунків зазвичай залишається поза межами дослідження, при тому, що складність та експлуатаційні навантаження є високими. В [51] представлена порівняльна характеристика роботи WebRTC застосунків в контейнеризованому оточенні Docker та віртуальній машині, що показало кращу ефективність використання ресурсів при використанні контейнеризації, аніж віртуалізації.

В [52] здійснено огляд основних відкритих джерел SFU серверів (Jitsi, Janus, Mediasoup, Medooze) та використання інструментів автоматичного тестування, що є корисним для проведення майбутніх експериментів. На основі результатів цієї роботи було вибрано SFU сервер Jitsi для огляду розгортання. Питання концепції Cloud Native розглянуто в [53], а в [54] – доцільність міграції застосунків відповідно цьому патерну, оскільки в деяких випадках це економічно невигідно.

Однак у проаналізованих роботах не досліджено питання спрощення експлуатації та розгортання SFU серверів в Cloud Native екосистемі. Тому такі дослідження є актуальними і доцільними з точки зору комерційного застосування та подальших експериментів. За рахунок автоматизації очікується зменшення часу на підняття інфраструктурних компонентів та зниження впливу людського фактору, що покращить якісну та кількісну складову потенційних експериментів у цій сфері.

1.6 Мета і задачі дисертаційного дослідження

Проведений аналіз показав необхідність розгляду концепції MLOps в розрізі інструментів Kubeflow із використанням низки визначених факторів покращення, зокрема таких як швидкість розробки, зменшення кількості помилок в моделі, часу на пошук проблем тощо. Можливості сучасних MLOps рішень можуть сприяти покращенню процесів розробки інформаційних систем МН.

Проведений аналіз показав необхідність продовження дослідження можливостей сучасних MLOps рішень для сприяння покращення процесів

розробки інформаційних систем МН. Висока портативність Kubernetes та Kubeflow дозволяє розгортати додатки на різних систем, тому доцільно розглянути концепції системи на базі дистрибутивів Kubernetes, та можливість встановлення на мобільних агентах. Також для підвищення функціональної стійкості розподіленої інформаційної системи мобільних агентів доцільно розглянути використання бездротових мереж на кшталт Mesh-мереж.

Для досягнення поставленої в роботі мети потрібно вирішити наступні окремі завдання дослідження:

дослідити можливості використання сучасних MLOps рішень для покращення процесів розробки інформаційних систем МН в розрізі інструментів Kubeflow;

розробити модель побудови та застосування МН із можливістю постійного донавчання, що досягається високим рівнем інтеграції та автоматизації конвеєру із використанням компонентів Kubeflow та платформи Kubernetes;

експериментально перевірити результати експлуатації моделі в арсеналі Kubeflow із використанням таких факторів покращення як швидкість розробки, імплементації змін, зменшення часу на пошук проблем, відновлення після глобальних перебоїв, зменшення кількості помилок в моделі;

розробити архітектурну концепцію системи на базі дистрибутива Kubernetes k3s з використанням моделі машинного навчання в парадигмі граничних обчислень (Edge computing) для мобільних агентів із використанням mesh-мереж;

провести оцінку ефективності використання бездротової Mesh-мережі для підвищення функціональної стійкості розподіленої інформаційної системи БПлА.

1.7 Список використаних джерел

1. Ethem A., “Introduction to Machine Learning”. MIT, vol. XIX, 2020, no. 1–3, pp. 13–18.

2. Hu J., Niu H., Carrasco J., Lennox B., Arvin F., “Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning”. IEEE Transactions on Vehicular Technology, vol. 69, 2020, no. 12, pp. 14413–14423.

3. Y.-N. Mohsen, Hugh E., Tulpan D., Sulik J., Eskandari M., “Application of Machine Learning Algorithms in Plant Breeding: Predicting Yield, Hyperspectral Reflectance in Soybean?”. Front. Plant Sci., 2021, no. 11, 624273.

4. Friedman J. H., “Data Mining and Statistics: What’s the connection?”. Computing Science and Statistics, vol. 29, 1998, no. 1, pp 3–9.

5. Kshetri N., “1 Blockchain’s roles in meeting key supply chain management objectives”. International Journal of Information Management, vol. 39, 2018, pp. 80–89.

6. L. Jiang, Z. Zhou, T. Leung, L.-J. Li, L. Fei-Fei, “MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels”. Proceedings of the 35th International Conference on Machine Learning, PMLR 80, 2018, pp. 2304-2313.

7. Y. Xiao, S. Zheng, J. Shi, X. Du, J. Hong. Knowledge graph-based manufacturing process planning: A state-of-the-art review, Journal of Manufacturing Systems, vol. 70, 2023, pp. 417–435/

8. Bozinovski S., “A self-learning system using secondary reinforcement”. In Trappl, Robert (ed.). Cybernetics and Systems Research: Proceedings of the Sixth European Meeting on Cybernetics and Systems Research. North-Holland, 1982, pp. 397–402.

9. Bozinovski S., “Modeling mechanisms of cognition-emotion interaction in artificial neural networks, since 1981”. Procedia Computer Science, 2014, pp. 255–263.

10. Синєглазов В., Чумаченко О., “Методи та технології напівкерованого навчання: Курс лекцій”. Київ: НТУУ “КПІ ім. Ігоря Сікорського”, 2022.

11. Дуда О. М., Кунанець Н. Е., Мацюк О. В., Пасічник В. В., “Методи аналітичного опрацювання big data”. Інтелектуальні системи прийняття рішень

та проблеми обчислювального інтелекту. Залізний Порт (21–27 травня 2018), с. 159.

12. Кропивницька В. Б., Магас Д. М., “Напівкероване машинне навчання для виявлення несправностей нафтогазопроводів”. Modern engineering and innovative technologies (30 квітня 2023), 1 (18), с. 33–36.

13. Russell S. J., Norvig P., “Artificial Intelligence: A Modern Approach”. Third edition. Prentice Hall, 2010.

14. Mohri M., Rostamizadeh A., Talwalkar A., “Foundations of Machine Learning”. The MIT Press, 2012.

15. Хорошилов С. В., Редька М. О., “Інтелектуальне керування орієнтацією космічних апаратів із використанням навчання з підкріпленням”. Техн. механіка, 2019, № 4, с. 29–43.

16. Van Otterlo M., Wiering M., “Reinforcement learning and Markov decision processes. Reinforcement Learning”. Adaptation, Learning, and Optimization, 2012, vol. 12. pp. 3–42.

17. Schulz H., Behnke S., “Deep Learning. Künstl Intell”, vol 26, 2012, pp. 357–363.

18. T. Hastie, R. Tibshirani and J. H. Friedman, “The Elements of Statistical Learning”. Springer. 2001.

19. Хома Ю. В., Бенч А. Я., “Порівняльний аналіз програмно-апаратного забезпечення алгоритмів глибокого навчання”. Комп’ютерні системи і мережі, 2019, № 1 (1), с. 97–102.

20. Бродкевич В. М., Ремесло В. Я., “Алгоритми машинного навчання (МН) та глибокого навчання (ГН) і їх використання в прикладних додатках”. Міжнародний науковий журнал «Інтернаука», 2018, т. 1, № 11 (51), с. 56–60.

21. LeCun Y., Bengio Y. and Hinton G., “Deep learning”. Nature, 521, 2015, pp. 436–444.

22. Krizhevsky A., Sutskever I., Hinton G., “ImageNet Classification with Deep Convolutional Neural Networks”. NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada, pp. 1–9.

23. Rosenblatt F., “Principles of Neurodynamics”. Spartan, New York, 1962.

24. Bhadeshia H. K. D. H., “Neural Networks in Materials Science”. ISIJ International, vol. 39, 1999, No. 10, pp. 966–979.

25. Zhang W., “Parallel distributed processing model with local space-invariant interconnections and its optical architecture”. Applied Optics, vol. 29, 1990, No.32, pp. 4790–4797.

26. M. Aykol, P. Herring and A. Anapolsky, “Machine learning for continuous innovation in battery technologies”. Nat. Rev. Mater, vol. 5, 2020, No. 10, pp. 725–727.

27. M. K. Gourisaria, R. Agrawal, G. M. Harshvardhan, M. Pandey and S. S. Rautaray. Application of Machine Learning in Industry 4.0. Machine Learning: Theoretical Foundations and Practical Applications. Springer. 2021, pp. 57–87.

28. A. De Las Heras, A. Luque-Sendra and F. Zamora-Polo. Machine learning technologies for sustainability in smart cities in the post-covid era. Sustainability, vol. 12, 2020, No. 22, p. 9320.

29. Google Cloud. MLOps: Continuous Delivery and Automation Pipelines in Machine Learning. [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (дата звернення 01.11.2023).

30. D. Kreuzberger, N. Kühl and S. Hirschl. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. Computer Science: Machine Learning, Vol. 11, 2023, p. 31866.

31. J. J. Boutilier and T. C. Y. Chan. Introducing and Integrating Machine Learning in an Operations Research Curriculum: An Application-Driven Course. INFORMS Transactions on Education, Vol. 23, 2023, No. 2, pp. 64–83.

32 Fiebrink R. Machine Learning Education for Artists, Musicians, and Other Creative Practitioners. ACM Transactions on Computing Education, Vol. 19, 2019, No. 4, Art. No. 31, pp. 1–32.

33. Sulmont E., Patitsas E. and Cooperstock J. R. What Is Hard about Teaching Machine Learning to Non-Majors? Insights from Classifying Instructors’ Learning

Goals. ACM Transactions on Computing Education, Vol. 19, 2019, No. 4, Art. No. 33, pp. 1–16.

34. Sachez-Pena M., Vieira C. and Magana A. Data science knowledge integration: Affordances of a computational cognitive apprenticeship on student conceptual understanding. Computer Applications in Engineering Education, vol. 31, 2022, pp. 239–259.

35. Elmachtoub A. N., Grigas P. Smart "Predict, then Optimize." Management Science, vol. 68, 2022, No. 1, pp. 9–26.

36. Mišić V. V. and Perakis G. Data analytics in operations management: A review. Manufacturing Service Operations Management. INFORMS, vol. 22, 2020, No. 1, pp. 158–169.

37. Hazzan O., Mike K. Core Concepts of Machine Learning. Guide to Teaching Data Science. 2023, pp. 209-222.

38. R. Chen, Y. Pu, B. Shi, W. Wu. An automatic model management system and its implementation for AIOps on microservice platforms. The Journal of Supercomputing, vol. 79, 2023, pp. 11410–11426.

39. H. Cai, C. Wang and X. Zhou. Deployment and verification of machine learning tool-chain based on kubernetes distributed clusters. CCF Transactions on High Performance Computing, vol. 3, 2021, pp. 157–170.

40. J. Moeyersons, B. Stamper, B. Volckaert and F. De Turck. Implementing a Network-Aware Kubernetes Scheduler on top of a Mesh Network, 2022 IEEE Cloud Summit, Fairfax, VA, USA, 2022, pp. 75–82.

41. Яровий, О. В., Завгородній, В. В. Розробка системи управління мобільними агентами, Таврійський науковий вісник, серія: Технічні науки, 2023 № 3, с. 29–36.

42. C. Yang, W. Wang, Y. Zhang, Z. Zhang, L. Shen. MLife: a lite framework for machine learning lifecycle initialization. Machine Learning, vol. 110, 2021, pp. 2993–3013.

43. Heater B. Twitter says staff can continue working from home permanently – TechCrunch. TechCrunch. [Електронний ресурс] – Режим доступу до ресурсу:

https://techcrunch.com/2020/05/12/twitter-says-staff-can-continue-working-from-home-permanently/ (дата звернення 01.11.2023).

44. Кузьміч М. Ю., Гордієнко Т. Б., “Імплементація та автоматизація розгортання WebRTC застосунків в Cloud Native оточенні”. Телекомунікаційні та інформаційні технології, 2021, вип. 3 (72), с. 54–62.

45. Kuzmich M. Ю., Zolotuhina O. A., “Analysis of challenges faced by implementation webrtc sfu gateway in cluster computing environment”. Збірник матеріалів Науково-технічної конференції “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”, Київ: ДУТ, 2020, с. 58–60.

46. Кузьміч М. Ю., “Особливості імплементація WebRTC застосунків в cloud native оточенні”. Збірник матеріалів науково-практичної конференцію “Проблеми комп’ютерної інженерії”, Київ: ДУТ, 2020, с. 23–25.

47. Pasha M., Shahzad F., Ahmad A. Analysis of challenges faced by WebRTC videoconferencing and a remedial architecture. International Journal of Computer Science and Information Security, Vol. 14, 2016, No. 10, pp. 698–705.

48. M. Antunes, C. Silva and J. Barranca. Telemedicine application using webrtc. Procedia computer science, vol. 100, 2016, pp. 414–420.

49. P. Pierleoni, L. Pernini et all. An innovative WebRTC solution for e-Health services, IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), Munich, 14 September 2016, 16484380.

50. Kwok-Fai N., Yang L., Wu C. A P2P-MCU Approach to Multi-Party Video Conference with WebRTC, International Journal of Future Computer and Communication, vol. 3, 2014, No. 5, pp. 319–324.

51. C. C. Spoiala et al. Performance comparison of a WebRTC server on Docker versus virtual machine, International Conference on Development and Application Systems (DAS), Suceava, 19 April 2016.

52. E. André et al. Comparative Study of WebRTC Open Source SFUs for Video Conferencing, Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, 18 October 2018, 18324713.

53. Gannon D., Barga R., Sundaresan N. Cloud-Native Applications. IEEE Cloud Computing, vol. 5, 2017, no. 4, pp. 16–21.

54. D. Linthicum. Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between, IEEE Cloud Computing, vol. 5, 2017, no. 4, pp. 12–14.

РОЗДІЛ 2
ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ Kubernetes ТА Kubeflow У КОНТЕКСТІ MLOps

2.1. Загальний огляд технології Kubernetes

Технологія Kubernetes розроблена компанією Google і підтримує низку інструментів з управління контейнерами. Її назва походить з грецької та означає керманич або пілот. У 2014 році Google відкрив доступ до вихідного коду проекту Kubernetes, який побудовано на базі п’ятнадцятирічного досвіду компанії [1].

Вона є рушієм оркестрації контейнерів з відкритим вихідним кодом для автоматичного розгортання, масштабування і управління контейнеризованими застосунками. Цей проект розроблявся під егідою Фонду природних обчислень CNCF (Cloud Native Computing Foundation). Це платформа з відкритим вихідним кодом для управління контейнеризованими робочими навантаженнями та супутніми службами. Її основні характеристики такі: кросплатформеність; розширюваність; успішне використання декларативної конфігурації та автоматизації. Вона має гігантську і швидкопрогресуючу екосистему [1–4].

Kubernetes не є комплексною системою «Платформа як послуга» (PaaS) у традиційному розумінні. Оскільки вона оперує швидше на рівні контейнерів, аніж на рівні апаратного забезпечення, деяка загальнозастосована функціональність і справді є спільною з PaaS як розгортання, масштабування, розподіл навантаження, логування і моніторинг. Водночас Kubernetes не є монолітним, а вищезазначені особливості підключаються і є опціональними. Kubernetes надає будівельні блоки для створення платформ для розробників, але залишає за користувачем право вибору у важливих питаннях.

Особливості Kubernetes такі:

1. Не обмежує типи застосунків, що підтримуються. Вона намагається підтримувати найрізноманітніші типи навантажень, включно із застосунками зі станом (stateful) та без стану (stateless), навантаження по обробці даних тощо.

Якщо ваш застосунок можна контейнеризувати, він чудово запуститься під Kubernetes.

2. Не розгортає застосунки з вихідного коду та не збирає ваші застосунки. Процеси безперервної інтеграції, доставки та розгортання (CI/CD) визначаються на рівні організації, та в залежності від технічних вимог.

3. Не надає сервіси на рівні застосунків як вбудовані: програмне забезпечення проміжного рівня (наприклад, шина передачі повідомлень), фреймворки обробки даних (наприклад, Spark), бази даних (наприклад, MySQL), кеш, некластерні системи збереження інформації (наприклад, Ceph). Ці компоненти можуть бути запущені у Kubernetes та/або бути доступними для застосунків за допомогою спеціальних механізмів, наприклад Open Service Broker.

4. Не нав’язує використання інструментів для логування, моніторингу та сповіщень, натомість надає певні інтеграційні рішення як прототипи, та механізми зі збирання та експорту метрик.

5. Не надає та не змушує використовувати якусь конфігураційну мову/систему (як наприклад Jsonnet), натомість надає можливість використовувати API, що може бути використаний довільними формами декларативних специфікацій.

6. Не надає і не запроваджує жодних систем машинної конфігурації, підтримки, управління або самозцілення.

7. Це не просто система оркестрації, так як вона усуває потребу оркестрації як такої. Технічним визначення оркестрації є запуск визначених процесів: спочатку A, за ним B, потім C. На противагу, Kubernetes складається з певної множини незалежних, складних процесів контролерів, що безперервно опрацьовують стан у напрямку, що заданий бажаною конфігурацією. Неважливо, як саме дістатися з пункту A до пункту C. Централізоване управління також не є вимогою. Все це виливається в систему, яку легко використовувати, яка є потужною, надійною, стійкою та здатною до легкого розширення.

Еволюція доставки програмних застосунків зображена на рис. 2.1.

Рис. 2.1 Еволюція доставки програмних застосунків

1. *Ера традиційного розгортання*. На початку організації запускали застосунки на фізичних серверах. Оскільки в такий спосіб не було можливості задати обмеження використання ресурсів, це спричиняло проблеми виділення та розподілення ресурсів на фізичних серверах. Наприклад, якщо багато застосунків було запущено на фізичному сервері, могли траплятись випадки, коли один застосунок забирав собі найбільше ресурсів, внаслідок чого інші програми просто не справлялись з обов’язками. Рішенням може бути запуск кожного застосунку на окремому фізичному сервері. Але такий підхід погано масштабується, оскільки ресурси не повністю використовуються; на додачу, це дорого, оскільки організаціям потрібно опікуватись багатьма фізичними серверами.

2. *Ера віртуалізованого розгортання*. Як рішення попередньої проблеми була представлена віртуалізація. Вона дозволяє запускати численні віртуальні машини (ВМ, virtual machine) на одному фізичному центральному процесорі сервера. Віртуалізація дозволила застосункам бути ізольованими у межах віртуальних машин та забезпечувала безпеку, оскільки інформація застосунку на одній ВМ не була доступна застосунку на іншій ВМ. Віртуалізація забезпечує краще використання ресурсів на фізичному сервері та кращу масштабованість, оскільки дозволяє легко додавати та оновлювати застосунки, зменшує витрати

на фізичне обладнання тощо. З віртуалізацією можна представити ресурси у вигляді одноразових ВМ. Кожна ВМ є повноцінною машиною з усіма компонентами, включно з власною операційною системою, що запущені поверх віртуалізованого апаратного забезпечення.

3. *Ера розгортання контейнерів*. Контейнери схожі на ВМ, але мають спрощений варіант ізоляції і використовують спільну операційну систему для усіх застосунків. Саму тому контейнери вважаються легковісними. Подібно до ВМ, контейнер має власну файлову систему, центральний процесор, пам’ять, простір процесів тощо. Оскільки контейнери вивільнені від підпорядкованої інфраструктури, їх можна легко переміщати між хмарними провайдерами чи дистрибутивами операційних систем.

Контейнери стали популярними, бо надавали додаткові переваги, такі як:

1. Створення та розгортання застосунків за методологією Agile: спрощене та більш ефективне створення образів контейнерів у порівнянні до використання образів віртуальних машин.

2. Безперервна розробка, інтеграція та розгортання: забезпечення надійних та безперервних збирань образів контейнерів, їх швидке розгортання та легкі відкатування (за рахунок незмінності образів).

3. Розподіл відповідальності команд розробки та експлуатації: створення образів контейнерів застосунків під час збирання/релізу на противагу часу розгортання, і як наслідок, вивільнення застосунків із інфраструктури.

4. Спостереження не лише за інформацією та метриками на рівні операційної системи, але й за станом застосунку та іншими сигналами.

5. Однорідність середовища для розробки, тестування та робочого навантаження: запускається так само як на робочому комп’ютері, так і у хмарного провайдера.

6. Операційна система і хмарна кросплатформність: запускається на Ubuntu, RHEL, CoreOS, у власному дата-центрі, у Google Kubernetes Engine і взагалі будь-де.

7. Керування орієнтоване на застосунки: підвищення рівня абстракції від запуску операційної системи у віртуальному апаратному забезпеченні до запуску застосунку в операційній системі, використовуючи логічні ресурси.

8. Нещільно зв'язані, розподілені, еластичні, вивільнені мікросервіси: застосунки розбиваються на менші, незалежні частини для динамічного розгортання та управління, на відміну від монолітної архітектури, що працює на одній великій виділеній машині.

9. Ізоляція ресурсів: передбачувана продуктивність застосунку.

10. Використання ресурсів: висока ефективність та щільність.

Зважаючи на представлені переваги контейнерів, вони потребують додаткової уваги при дисертаційних дослідженнях.

2.2 Основні характеристики і компоненти Kubernetes

2.2.1 Основні характеристики Kubernetes

Контейнери є прекрасним способом упаковки і запуску застосунків. Потрібно керувати контейнерами, в яких працюють застосунки, і стежити, щоб не було простою [5–7]. Наприклад, якщо один контейнер припиняє роботу, інший має бути запущений йому на заміну. Kubernetes надає каркас для еластичного запуску розподілених систем. Він опікується масштабуванням та аварійним відновленням застосунку, пропонує шаблони розгортань тощо. Наприклад, Kubernetes дозволяє легко створювати розгортання за стратегією canary.

Основні характеристики Kubernetes такі:

1. *Виявлення сервісів та балансування навантаження*. Kubernetes може надавати доступ до контейнера, використовуючи DNS-ім'я або його власну IP-адресу. Якщо контейнер зазнає завеликого мережевого навантаження, то Kubernetes здатний збалансувати та розподілити його таким чином, щоб якість обслуговування залишалась стабільною.

2. *Оркестрація сховища інформації*. Kubernetes дозволяє автоматично монтувати системи збереження інформації на ваш вибір: локальні сховища, рішення від хмарних провайдерів тощо.

3. *Автоматичне розгортання та відкатування*. За допомогою Kubernetes можна описати бажаний стан контейнерів, що розгортаються, і регульовано простежити за виконанням цього стану. Наприклад, можна автоматизувати процеси створення нових контейнерів для розгортання, видалення існуючих контейнерів і передачу їхніх ресурсів на новостворені контейнери.

4. *Автоматичне розміщення задач*. Kubernetes надається кластер для запуску контейнерізованих задач і вказується, скільки ресурсів центрального процесора та оперативної пам’яті необхідно для роботи кожного контейнера. Kubernetes розподіляє контейнери по вузлах кластера для максимально ефективного використання ресурсів.

5. *Самозцілення*. Kubernetes перезапускає контейнери, що відмовили; заміняє контейнери; зупиняє роботу контейнерів, що не відповідають на задану користувачем перевірку стану, і не повідомляє про них клієнтам, допоки ці контейнери не будуть у стані робочої готовності.

6. *Управління секретами та конфігурацією*. Kubernetes дозволяє зберігати та керувати чутливою інформацією, такою як паролі, OAuth (скорочення від Open Authorization) токени і SSH (Secure Shell – «безпечна оболонка») ключі. Можна розгортати та оновлювати секрети та конфігурацію без перезбирання образів ваших контейнерів, не розкриваючи секрети в конфігурацію стека.

При розгортанні Kubernetes отримується кластер, який складається з набору робочих машин, званих вузлами, які запускають контейнерезовані програмні застосунки. Кожен кластер має щонайменше один робочий вузол. На робочому вузлі розміщуються Pod – основна одиниця планувальника в Kubernetes. Панель управління керує робочими вузлами і Pod в кластері. У виробничих середовищах панель управління зазвичай розподілена на декількох серверах, а кластер зазвичай працює на декількох вузлах, забезпечуючи стійкість до відмов і високу доступність.

2.2.2 Основні компоненти Kubernetes

У рис. 2.2 наведені основні компоненти, необхідні для того, щоб мати повний та працюючий кластер Kubernetes [5–7].

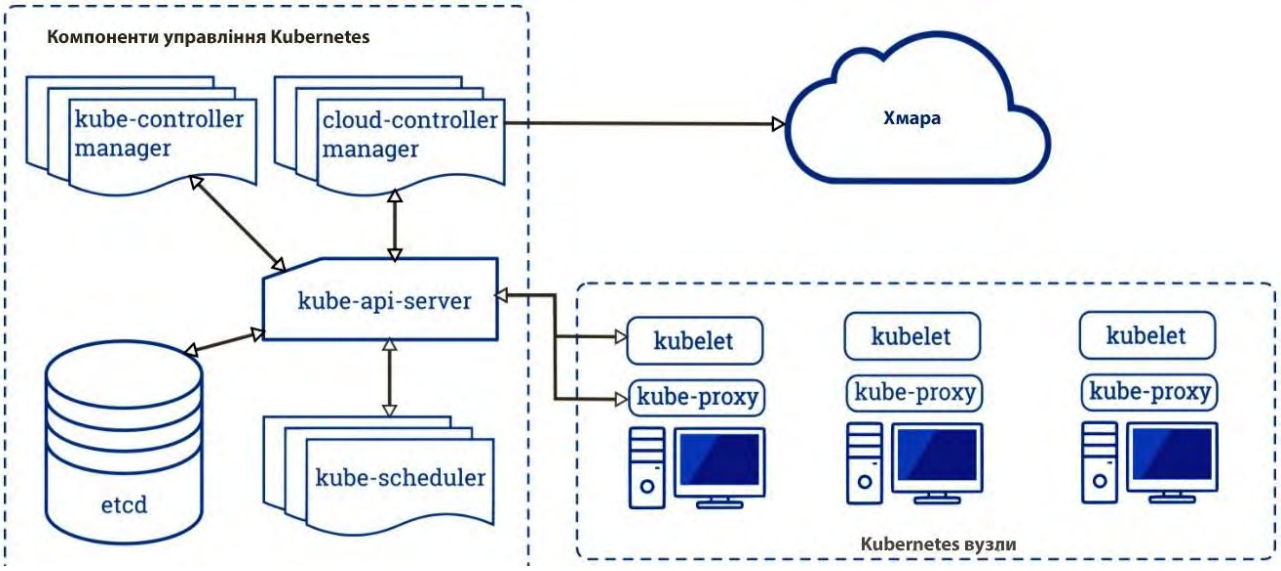


Рис. 2.2 Схема кластеру Kubernetes з усіма її компонентами

Розглянемо компоненти панелі управління, які приймають глобальні рішення щодо кластера (наприклад, планування), а також виявляють і реагують на події кластера (наприклад, запуск нового струмка, коли поле реплік розгортання не задоволено). Ці компоненти можуть працювати на будь-якій машині кластера. Однак для простоти налаштування сценаріїв зазвичай запускають усі компоненти панелі управління на одній машині та не запускають користувальницькі контейнери на цій машині.

Kube-apiserver – компонент панелі управління Kubernetes, який відкриває API Kubernetes. Сервер API – це вхідна точка перед панеллю управління Kubernetes. Основна реалізація сервера API Kubernetes – це Kube-apiserver. Kube-apiserver призначений для горизонтального масштабування, тобто масштабується, збільшуються кількість екземплярів. Можна запустити кілька екземплярів кубе-apiserver і збалансувати трафік між ними.

Etcd – це консистентне (цілісне) і високодоступне сховище ключ-значення, що використовується як резервна копія Kubernetes для всіх даних кластера. Kube-scheduler – компонент панелі управління, який спостерігає за новоствореними Pod, і вибирає вузол, на якому вони будуть запущені.

Фактори, що враховуються для прийняття рішень щодо планування, включають: індивідуальні та колективні вимоги до ресурсів, обмеження в апаратному / програмному забезпеченні / політиці, специфічності спорідненості, локальність даних, робочим навантаженням.

Kube-controller-manager – це компонент панелі управління, який відповідає за роботу процесів контролера. За логікою, кожен контролер – окремий процес, але для зменшення складності всі вони складаються в єдиний бінарний і запускаються в один процес.

До таких контролерів належать:

1) *контролер вузла* – відповідає за мічення та реагування, коли вузли опускаються;

2) *контролер реплікації* – відповідає за підтримання правильної кількості Pod для кожного об’єкта контролера реплікації в системі;

3) *контролер API ендпоінтів*;

4) *контролери облікових записів та токенів* – створюють облікові записи за замовчуванням та API токени доступу для нових просторів імен.

Cloud-controller-manager – це компонент панелі управління Kubernetes, який вбудовує специфічну для хмари логіку управління. Менеджер хмарних контролерів дозволяє зв’язати ваш кластер з API вашого провайдера хмарних продуктів та відокремити компоненти, які взаємодіють із цією хмарною платформою, від компонентів, які просто взаємодіють із вашим кластером. Цей компонент запускає лише контролери, характерні для вашого постачальника хмар. При запуску Kubernetes у власних центрах даних або на локальному середовищі власного комп’ютера кластер не має менеджера хмарного контролера cloud-controller-manager.

Як і у випадку з kube-controller-manager, cloud-controller-manager поєднує кілька логічно незалежних циклів управління в єдиний бінарний файл, який ви запускаєте як єдиний процес. Можна масштабувати горизонтально (виконати більше однієї копії), щоб поліпшити продуктивність або збільшити надійність системи.

Розглянемо компоненти робочого вузла, які працюють на кожному робочому вузлі, підтримуючи запущені Pod і забезпечуючи роботу середовища Kubernetes.

Kubelet – це агент, який працює на кожному вузлі кластера. Це гарантує, що контейнери працюють у Pod. Він містить набір PodSpecs, які надаються за допомогою різних механізмів і забезпечують, щоб контейнери, описані в цих PodSpecs, працювали і були активні. Kubelet не керує контейнерами, які не були створені Kubernetes.

Kube-proxy – це мережевий проксі, який працює на кожному вузлі вашого кластера, реалізуючи частину концепції Kubernetes Service. Він підтримує мережеві правила щодо вузлів, які дозволяють здійснювати мережеве спілкування з вашими Pod в самому кластері так і за його межами. Kube-proxy використовує шар фільтрації пакетів операційної системи, якщо такий існує. В іншому випадку kube-proxy пересилає сам трафік без змін.

Container runtime – це програмне забезпечення, яке відповідає за запуск контейнерів. Kubernetes підтримує кілька режимів виконання контейнерів: Docker, containerd, CRI-O і будь-яку реалізацію інтерфейсу виконання контейнера Kubernetes CRI (Container Runtime Interface).

2.3 Контролери Kubernetes

Контролер активно моніторить та підтримує набір ресурсів Kubernetes у бажаному стані. Kubernetes складається із набору контролерів, які регулярно стежать і узгоджують поточний стан додатків із заявленим цільовим станом. Розглянемо, як використовувати цю основну концепцію для розширення платформи для наших потреб [5–7].

Kubernetes – це складна і всеосяжна платформа, яка надає багато можливостей. Однак, це оркестрова платформа загального призначення, яка не охоплює всіх випадків використання додатків. В той же час, вона пропонує точку розширення, де конкретні випадки використання можуть бути реалізовані поверх перевірених складових Kubernetes. Основне питання, що виникає, – це те, як розширити Kubernetes, не змінюючи і не порушуючи її, і як використовувати її можливості для певних задач.

Kubernetes базується на декларацiйному API, орієнтованому на ресурси. На відміну від імперативного підходу, декларативний підхід не вказує Kubernetes, як вона повинна діяти, а натомість описує, як має виглядати кінцевий стан. Наприклад, при збільшенні масштабу розгортання активно не створюється нові Pods. Натомість змінюються властивості реплік ресурсу розгортання через API Kubernetes на потрібне число.

Нові Pod створюються внутрішніми контролерами. Для кожної зміни статусу ресурсу (як-от зміна значення властивостей реплік Deployment) Kubernetes створює подію і передає її всім зацікавленим слухачам. Ці слухачі потім можуть реагувати, змінюючи, видаляючи або створюючи нові ресурси, що, в свою чергу, створює інші події, наприклад події, створені Pod. Потім ці події потенційно знову підбираються іншими контролерами, які виконують свої конкретні дії.

Весь цей процес також відомий як узгодження стану, де цільовий стан (кількість бажаних реплік) відрізняється від поточного стану (фактичні запущені екземпляри), і завдання контролера знову примирити і досягти бажаного цільового стану. Kubernetes по суті представляє розподілену систему керування станом. При наданні їй потрібного стану для екземпляра компонента вона намагається підтримувати цей стан, якщо щось зміниться.

Kubernetes постачає колекцію вбудованих контролерів, які керують її стандартними ресурсами, такими як ReplicaSets, DaemonSets, StatefulSets, Deployments або Services. Ці контролери працюють як частина диспетчера контролерів, який розгорнуто (як окремий процес або Pod) на мастер вузлі. Ці

контролери не знають один про одного і працюють у нескінченному циклі перевірки стану, стежать за своїми ресурсами за фактичним та бажаним станом та відповідно діють, щоб наблизити фактичний стан до бажаного стану.

Однак, окрім стандартних контролерів, архітектура, керована подіями Kubernetes, дозволяє нам підключати інші спеціальні контролери. Спеціальні контролери можуть додати додаткову функціональність подій зміни стану поведінки так само, як це роблять внутрішні контролери. Загальна характеристика контролерів полягає в тому, що вони реагують на події в системі для виконання своїх конкретних дій.

На високому рівні процес примирення складається з наступних основних етапів:

1) *спостереження* – інформація про фактичний стан на основі спостережень за подіями, виданими Kubernetes під час спостереження зміни ресурсів;

2) *аналіз* – визначення відмінності від бажаного стану;

3) *дія* – виконання операції з приведення фактичного до потрібного стану.

Наприклад, контролер ReplicaSet спостерігає за зміною ресурсу ReplicaSet, аналізує, скільки Pods потрібно запустити, та діє, подаючи визначення Pod на сервер API. Потім запуск Kubernetes відповідає за запуск запитуваного Pod на вузол.

На рис. 2.3 показано, як контролер реєструється як слухач подій для виявлення змін на керованих ресурсах. Він спостерігає за поточним станом і змінює його, викликаючи сервер API, щоб наблизитися до цільового стану (за необхідності) [7].

Контролери є частиною панелі управління Kubernetes, і вони також дозволять розширити платформу за допомогою власної поведінки. Більше того, вони перетворилися на стандартний механізм розширення платформи і дозволяють розширити можливості керувати життєвим циклом додатків. В результаті створено нове покоління більш досконалих контролерів, які мають назву «Оператори».

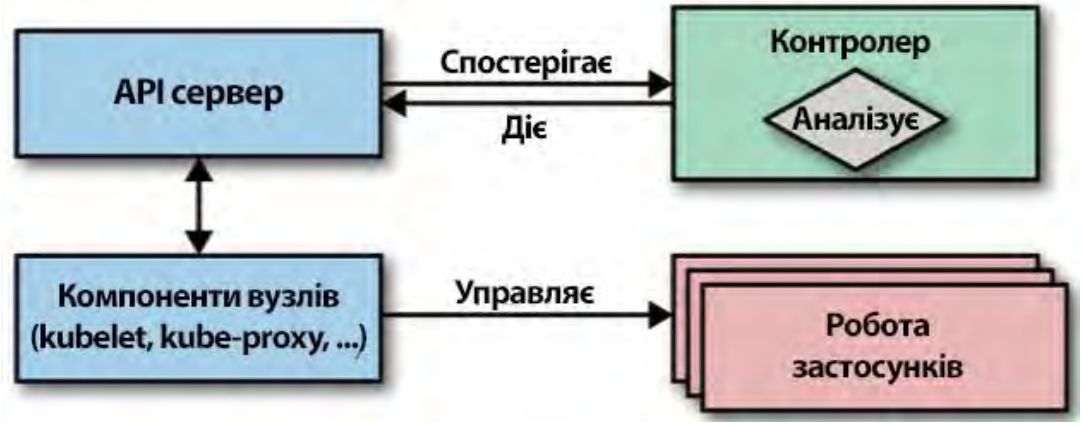


Рис. 2.3 Цикл спостереження-аналіз-дія

З точки зору еволюції та складності, можна класифікувати активні компоненти примирення на дві групи:

1) *контролери* – простий процес перевірким стану, який контролює та діє на стандартних ресурсах Kubernetes. Частіше ці контролери покращують поведінку платформи та додають нові функції платформи;

2) *оператори* – складний процес перевірки стану, який взаємодіє за допомогою спеціального визначення ресурсів CRD (Custom Resource Definitions), які лежать в основі схеми оператора. Зазвичай ці оператори інкапсулюють складну бізнес логіку додатків і керують повним життєвим циклом програми.

Ці класифікації допомагають вводити поступово нові поняття. Спочатку необхідно розглянути найпростіші контролери, а в наступному – CRD і патерн операторів.

Для уникнення наявності декількох контролерів, що діють на одних і тих же ресурсах одночасно, контролери використовують шаблон SingletonService. Більшість контролерів розгорнуто як Deployments, але з однією копією, оскільки Kubernetes використовує блокування на рівні ресурсу для запобігання проблем паралельності при зміні ресурсні об’єкти. Зрештою, контролер – це не що інше, як програма, яка постійно працює у фоновому режимі.

Оскільки Kubernetes написаний на Go, а повна клієнтська бібліотека для доступу до Kubernetes також написана в Go, багато контролерів також написані на Go. Однак, можна писати контролери на будь-якій мові програмування, надсилаючи запити на сервер API Kubernetes.

Найпростіший вид контролерів розширює спосіб управління Kubernetes з використанням ресурсів самої системи. Вони працюють на тих же стандартних ресурсах і виконують аналогічні завдання, що і внутрішні контролери Kubernetes, що працюють на стандартних ресурсах Kubernetes, але які невидимі для користувача кластери. Контролери оцінюють визначення ресурсів і умовно виконують деякі дії. Хоча вони можуть контролювати та діяти в будь-якому полі визначення ресурсу, метадані та ConfigMaps є найбільш підходящими для цієї мети.

Для вибору місця зберігання даних контролера необхідно розрізняти:

1. *Мітки* (label) – частини метаданих ресурсу може переглядати будь-який контролер. Вони індексуються в резервній базі даних і їх можна ефективно шукати в запитах. Необхідно використовувати мітки, коли потрібна функція, що нагадує селектор. Обмеження міток полягає в тому, що можна використовувати лише буквено-цифрові імена та значення з обмеженнями.

2. *Анотації* (annotation) – відмінна альтернатива міткам. Їх слід використовувати замість міток, якщо значення не відповідають синтаксичним обмеженням міток. Вони не індексуються, тому використовуються анотації для неопізнавальної інформації, яка не використовується як ключі контролера.

3. *Карти конфігурацій* (ConfigMaps). Іноді контролерам потрібна додаткова інформація, яка не вписується добре в мітки чи анотації. У цьому випадку ConfigMaps можна використовувати для зберігання кінцевого стану. Потім ці ConfigMaps переглядаються та читаються контролерами. Однак CRD набагато краще підходять для проектування спеціальних специфікацій кінцевого стану і рекомендуються над звичайними ConfigMaps. Однак, для реєстрації CRD потрібні підвищені дозволи на рівні кластера. Якщо таких немає, то ConfigMaps як і раніше є найкращою альтернативою для CRD.

Декілька досить простих прикладних контролерів, які можна визначити як зразок реалізації цього шаблону:

1) jenkins-x/exposecontroller – контролер, який спостерігає за значеннями Services, і якщо він виявляє в метаданих анотацію, названу expose, контролер автоматично прокидує об’єкт Ingress для зовнішнього доступу. Він також видаляє об’єкт Ingress, коли хтось видаляє Service;

2) fabric8/configmapcontroller – контролер, який спостерігає за об’єктами ConfigMap, за змінами і виконує оновлення пов’язаних з ними розгортань. Цей контролер можна використовувати з програмами, які не здатні сприймати ConfigMap і динамічно оновлювати себе новими конфігураціями. Це особливо вірно, коли Pod використовує цей ConfigMap як змінні середовища або коли програма не може швидко і надійно оновитись на ходу без перезавантаження;

3) оператор оновлення Linux Container Linux – контролер, який перезавантажує вузол Kubernetes, коли він виявляє певну анотацію на вузлі.

Наприклад: контролер, що складається з одного Bash сценарію, і спостерігає за API Kubernetes за змінами на ресурсах ConfigMap. Якщо помічається така ConfigMap за допомогою k8spatterns.io/podDeleteSelector, всі Pods, вибрані із заданим значенням анотації, видаляються, коли ConfigMap змінюється. Якщо припустити, що підтримуються ці Pods з ресурсомами, як Deployment або ReplicaSet, ці Pods перезапускаються та підтягують змінену конфігурацію.

2.3 Оператори Kubernetes

Оператором є контролер, який використовує CRD для інкапсулювання операційних знань для конкретного додатку в алгоритмічній та автоматизованій формі. Патерн операторів дозволяє розширити патерн для більшої гнучкості та більшої виразності [6, 8].

Контролер – це спосіб розширити платформу Kubernetes простим і розподіленим способом. Однак, для випадків розширеного використання звичайні користувацькі контролери недостатньо потужні, оскільки вони

обмежені лише переглядом та керуванням лише внутрішніми ресурсами Kubernetes. Іноді необхідно додати нові концепції до платформи Kubernetes, яка потребує додаткових об’єктів домену. Наприклад, при виборі Prometheus як свого моніторингового рішення виникає необхідність моніторити його чітко визначеним чином. Краще, щоб був ресурс Prometheus, який описував налаштування моніторингу і всі деталі розгортання, подібно до того, як визначаються інші ресурси Kubernetes.

Ці ситуації є типовими, коли CRD є дуже корисними. Вони дозволяють розширювати API Kubernetes, додаючи власні ресурси до вашого кластеру Kubernetes і використовуючи їх так, ніби вони є «рідними» ресурсами. Спеціальні ресурси разом із контролером, що діє на ці ресурси, формують шаблон оператора.

Дослідники при описі характеристик операторів, відзначають, що оператор – це контролер Kubernetes, який розуміє дві області: Kubernetes та щось інше. Поєднуючи знання обох галузей, він може автоматизувати завдання, які зазвичай вимагають від оператора людини, який розуміє обидві галузі. З допомогою контролера можна ефективно реагувати на зміни станів рідних ресурсів Kubernetes. Тепер, коли є розуміння одної половини шаблону оператора, то можна подивитись на другу половину – представлення власних ресурсів з використанням CRD.

За допомогою CRD можна розширити Kubernetes для управління доменними концепціями на платформі Kubernetes. Користувацькі ресурси керуються, як і будь-яким іншим ресурсом, через API Kubernetes і врешті-решт зберігаються в резервному сховищі Etcd. Історично попередниками CRD були ThirdPartyResources. Попередній сценарій фактично реалізований за допомогою цих нових користувацьких ресурсів оператором CoreOS Prometheus, щоб дозволити безперешкодну інтеграцію Prometheus до Kubernetes.

Приклад CRD наведений на рис. 2.4, а визначення підресурсів для CRD – на рис. 2.5 [7].

```
kind: CustomResourceDefinition
# ...
spec:
  subresources:
    status: {}
    scale:
      specReplicasPath: .spec.replicas ❶
      statusReplicasPath: .status.replicas ❷
      labelSelectorPath: .status.labelSelector ❸
```

Рис. 2.4 Приклад CRD

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: prometheuses.monitoring.coreos.com      ❶
spec:
  group: monitoring.coreos.com                  ❷
  names:
    kind: Prometheus                            ❸
    plural: prometheuses                        ❹
  scope: Namespaced                             ❺
  version: v1                                   ❻
  validation:
    openAPIV3Schema: ....                       ❼
```

❶ Ім'я

❷ API група, до якої належить ресурс

❸ Kind(тип) використовується для ідентифікації типу ресурсу

❹ Правило найменування множин ресурсів, використовується для їх сортування

❺ Область створення, може бути на рівні кластеру або namespace (ізольоване віртуальне оточення)

❻ Версія CRD

❼ OpenAPI V3 валідація схеми (не показана на рис.)

Рис. 2.5 Визначення підресурсів для CRD

Крім того, Kubernetes дозволяє вказувати два можливі субресурси для CRD за допомогою субресурсів у специфікаційному полі:

1. *Масштаб* (scale). За допомогою цього властивості CRD може вказати, як він керує своїм числом реплік. Це поле може бути використане для оголошення шляху JSON (JavaScript Object Notation), де вказана кількість бажаних реплік цього користувацького ресурсу. Шлях до властивості, що містить фактичну кількість запущених реплік і необов'язковий шлях до селектора, який можна використовувати для пошуку копії спеціальних екземплярів ресурсів. Цей селектор міток, як правило, необов'язковий, але необхідний, якщо ви хочете використовувати цей користувацький ресурс разом із HPA (HorizontalPodAutoscaler).

2. *Статус* (status). Коли встановити цю властивість, з'явиться новий виклик API, який дозволяє лише змінити статус. Цей виклик API може бути захищений індивідуально і дозволяє оновлювати статус поза контролером. З іншого боку, коли оновлюється спеціальний ресурс у цілому, розділ стану ігнорується, як і для стандартних ресурсів Kubernetes.

2.4 Методи дослідження роботи конвеєра машинного навчання

Під час роботи з МН визначають етапи, які можуть бути виконані як вручну, так і автоматично:

1. *Збір даних* – пошук і вибір релевантних даних із різних джерел для виконання задач МН.

2. *Аналіз даних* – проведення розвідувального аналізу – попередній експрес-аналіз даних шляхом їх перетворення та/або представлення у зручному вигляді: графічному, табличному, схемі, діаграмі тощо. Результатом буде наступне: розуміння структури та характеристик даних, що очікуються моделлю; визначення етапів підготовки даних та розробки основних особливостей (feature), необхідних для моделі.

3. *Обробка даних* – дані мають бути готові для виконання над ними задач МН. Відбувається їх очищення та розподіл на дані для тренування, валідацію та тестування.

4. *Тренування моделі* – інженери використовують різноманітні алгоритми та оброблені дані для тренування моделі. Додатково використовують алгоритми для підбору гіперпараметрів для покращення точності моделі.

5. *Оцінка моделі* – модель оцінюється на основі наявних даних. Результатом є набір метрик з якості моделі.

6. *Валідація моделі* – перевірка моделі на допустимість для розгортання в потрібне оточення.

7. *Експлуатація моделі* – перевірена модель розгорнута в кінцевому оточенні. Розгорнута модель може мати наступний вигляд: мікросервіс із REST (Representational State Transfer) API, що опрацьовує прогнози в режимі реального часу; вбудована модель в кінцевий пристрій; частина в складній системі прогнозування.

8. *Моніторинг моделі* – точність і швидкість прогнозування моделі вимірюється для врахування в наступних ітераціях розробки.

Рівень автоматизації цих процесів показує зрілість процесів МН, що впливає на швидкість розробки нових моделей. Можна виділити три основних рівні MLOps, починаючи з початкового рівня, який включає 0 автоматизації до рівня, із автоматизацію МН задач і повноцінного конвеєра неперервної інтеграції та розгортання. Найпростіший MLOps конвеєр зазвичай складається із таких етапів, як: підготовка даних, навчання моделі, валідація моделі та відповідно обслуговування/запуск кінцевої моделі.

Kubeflow – це Cloud Native відкритий ресурс-платформа (open-source) для МН, розроблена компанією Google [1, 2]. Особливістю цієї платформи є високий рівень інтеграції із системою оркестрації Kubernetes та великий набір інструментарію для задач МН. Kubeflow розробники організовані в робочі групи із відповідними репозиторіями, що фокусуються на специфічних ML задачах:

автоматичне машинне навчання; розгортання; маніфести – файли конфігурацій Kubernetes; Jupiter notebook; конвеєри; тренування.

Схематичне зображення конвеєра розробки рішень для МН у платформі Kubeflow зображено на рис. 2.6 (а – фаза експериментів Kubeflow; б – фаза експлуатації Kubeflow) [9, 10].

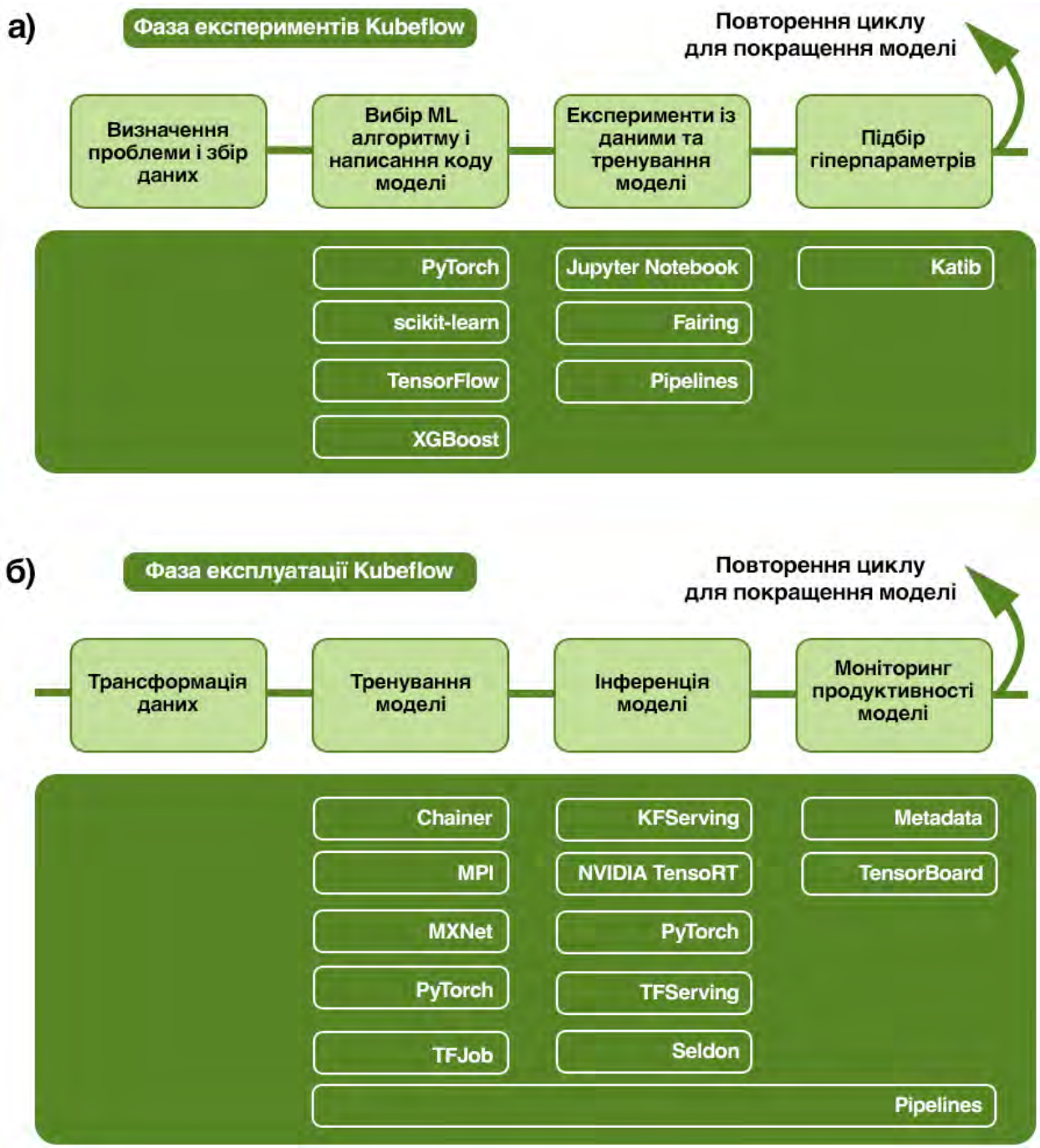


Рис. 2.6 Схематичне зображення конвеєра розробки рішень МН у Kubeflow

В дослідженні зроблено огляд основних компонентів Kubeflow і їх доцільність при створенні конвеєра. Для огляду обрано версію Kubeflow 1.12, яка була найновішою. Процес встановлення є відносно простий, оскільки створено застосунок інтерфейс командного рядка CLI (Command Line Interface) для цієї потреби. Основною функціональною задачею CLI застосунку є формування маніфестів для Kubernetes. Після інсталяції компоненти будуть доступні в просторі імен (namespace) Kubeflow. В платформі наявна велика кількість інтегрованих компонентів. Однак в роботі розглянуто тільки ті, що мінімально необхідні для побудови довершеного конвеєра.

Використання готового CLI застосунку значно спрощує процес інсталяції, оскільки дозволяє уникнути ручного редагування масивного конфігураційного файлу, зменшуючи поріг входу для користувачів. Список інстальованих компонентів Kubeflow зображено на рис. 2.7 [9, 10].

NAME	READY	STATUS	RESTARTS	AGE
admission-webhook-bootstrap-stateful-set-0	1/1	Running	0	8m24s
admission-webhook-deployment-5d9ccb5696-b4k2n	1/1	Running	0	8m21s
alb-ingress-controller-7dcf69ff67-w4wxl	1/1	Running	0	9m4s
application-controller-stateful-set-0	1/1	Running	0	8m53s
argo-ui-684bcb587f-rq4j9	1/1	Running	0	9m4s
cache-deployer-deployment-6667847478-p5l6t	2/2	Running	1	9m4s
cache-server-bd9c859db-svhq8	2/2	Running	0	9m3s
centraldashboard-895c4c768-r4ssh	1/1	Running	0	9m3s
jupyter-web-app-deployment-54758f7bc4-g2svl	1/1	Running	0	9m3s
katib-controller-75c8d47f8c-9nr9c	1/1	Running	0	9m3s
katib-db-manager-6c88c68d79-fbnzt	1/1	Running	0	9m3s
katib-mysql-858f68f588-zz7dn	1/1	Running	0	9m3s
katib-ui-68f59498d4-prl5s	1/1	Running	0	9m3s
kfserving-controller-manager-0	2/2	Running	0	8m44s
kubeflow-pipelines-profile-controller-69c94df75b-zfwff	1/1	Running	0	9m2s
metacontroller-0	1/1	Running	0	8m34s
metadata-db-757dc9c7b5-6xvjw	1/1	Running	0	9m2s
metadata-envoy-deployment-6ff58757f6-dkj64	1/1	Running	0	9m2s
metadata-grpc-deployment-76d69f69c8-bwp8b	1/1	Running	2	9m2s
metadata-writer-6d94ffb7df-7d4lg	2/2	Running	0	9m2s
minio-66c9cd74c9-n6659	1/1	Running	0	9m2s
ml-pipeline-54989c9946-m6gq8	2/2	Running	0	9m1s
ml-pipeline-persistenceagent-7f6bf7646-x29rq	2/2	Running	0	9m1s
ml-pipeline-scheduledworkflow-66db7bcf5d-5sb6f	2/2	Running	0	9m1s
ml-pipeline-ui-756b58fb-tcvtr	2/2	Running	0	9m1s
ml-pipeline-viewer-crd-58f59f87db-g29f9	2/2	Running	1	9m
ml-pipeline-visualizationserver-6f9ff4974-jgcwr	2/2	Running	0	9m
mpi-operator-77bb5d8f4b-68z2f	1/1	Running	0	9m
mxnet-operator-68b688bb69-xrr24	1/1	Running	0	9m
mysql-7694c6b8b7-ghbhv	2/2	Running	0	9m
notebook-controller-deployment-58447d4b4c-ndqkt	1/1	Running	0	9m
nvidia-device-plugin-daemonset-dt792	2/2	Running	0	8m29s
nvidia-device-plugin-daemonset-jbw9z	2/2	Running	0	8m38s
profiles-deployment-78d4549cbc-n45cx	2/2	Running	0	8m59s
pytorch-operator-b79799447-2k6t2	1/1	Running	0	8m59s
seldon-controller-manager-5fc5dfc86c-lht2f	1/1	Running	0	8m59s
spark-operatorsparkoperator-67c6bc65fb-7rn7f	1/1	Running	0	8m59s
tf-job-operator-5c97f4bf7-8zr5z	1/1	Running	0	8m59s
workflow-controller-5c7cc7976d-5t8m2	1/1	Running	0	8m59s

Рис. 2.7 Список інстальованих компонентів Kubeflow

Простий цикл розробки, де всі дії виконуються вручну, можна розділити на такі чотири етапи: обробка даних; тренування моделі; оцінка та валідація моделі; запуск моделі.

Розглянемо процес автоматизації цього конвеєра з допомогою платформи Kubeflow покомпонентно, оскільки якісний конвеєр передбачає наявність певного рівня ізоляції між окремими етапами і чітке розуміння, що приходить на вхід та буде на виході.

Jupyter Notebook – це open-source веб-додаток, який дозволяє створювати та обмінюватися документами, що містять робочий код, рівняння, візуалізації та текст. Notebook використовують для очищення та перетворення даних, чисельного моделювання, статистичного моделювання, візуалізації даних, МН та багато іншого. Спектр використання jupiter notebook є дуже широким і використовується на всіх етапах конвеєра.

Зберігання метаданих про результати навчання є важливим етапом розробки систем МН, що дозволяє оцінити різні версії моделі, порівнювати їх між собою. Проєкт Metadata, який є аналогом MLFlow, створений для того, аби вирішити проблеми користувачів із розумінням їх процесів МН, відслідковуючи і опрацьовуючи метадані, які створені цим процесом. В цьому контексті метадані означають інформацію про моделі, дата-сет (набір даних) та інші артефакти. Артефакти це файли, які формують вхід та вихід між різними етапами конвеєра.

В цьому випадку Metadata дозволяє відповісти на такі питання:

1. На якому датасеті тренувалась модель?
2. Які гіперпараметри використовувались для тренування моделі?
3. До якої ітерації тренування належить модель?
4. На якій версії TensorFlow створювалась модель?
5. Коли було завантажено неробочу модель?

Тренування моделі це процес оптимізації алгоритму залежно від вхідних даних. Kubeflow має в собі оператори, які автоматизують навчання для таких фреймворків МН як: MXNet; PyTorch; TensorFlow.

Концепція Kubernetes операторів дозволяє спростити процес навчання та представити процес як окремий kubernetes об’єкт. Складність процесу схована за додатковою абстракцією і, як результат, відносно складна задача розподіленого МН може бути представлена одним kubernetes об’єктом та описана YAML файлом маніфестом.

Налаштування гіперпараметрів (AutoML) – це задача МН з вибору множини оптимальних гіперпараметрів для алгоритму МН. Katib – проєкт для розробки систем МН, що підтримує оптимізацію гіперпараметрів, пошук нейронної архітектури NAS (Neural Architecture Search). Katib дозволяє задати швидкість навчання, кількість шарів у нейронній мережі та кількість вузлів у кожному шарі.

Kubeflow pipeline є ключовим елементом розробки системи МН, об’єднуючи всі компоненти МН в повноцінний конвеєр (рис. 2.8) [9]. Перевагою Kubeflow pipeline у порівнянні із іншими системами є наявність набору засобів розроблення SDK (Software Development Kit) на мові програмування Python, яка покриває основні Kubeflow абстракції. Описання конвеєра розробки тією ж мовою програмування, що і код моделі, позитивно впливає на процес, оскільки руйнується абстрактна стіна між розробкою та експлуатацією. Розробкою коду та автоматизацією процесів може займатись один інженер.

Kubeflow Pipelines SDK включає наступні пакети, основні з них:

kfp.compiler – включає класи і методи для компіляції конвеєра написаному на Python DSL в маніфести YAML, що застосовуються в Kubernetes;

kfp.components – включає класи і методи для взаємодії із різними компонентами;

kfp.dsl – включає предметно-орієнтовану мову програмування DSL (Domain-Specific Language), що використовується для взаємодії із конвеєром та його компонентами;

kfp.Client, включає Python клієнтські бібліотеки для взаємодії із Kubeflow Pipelines API,

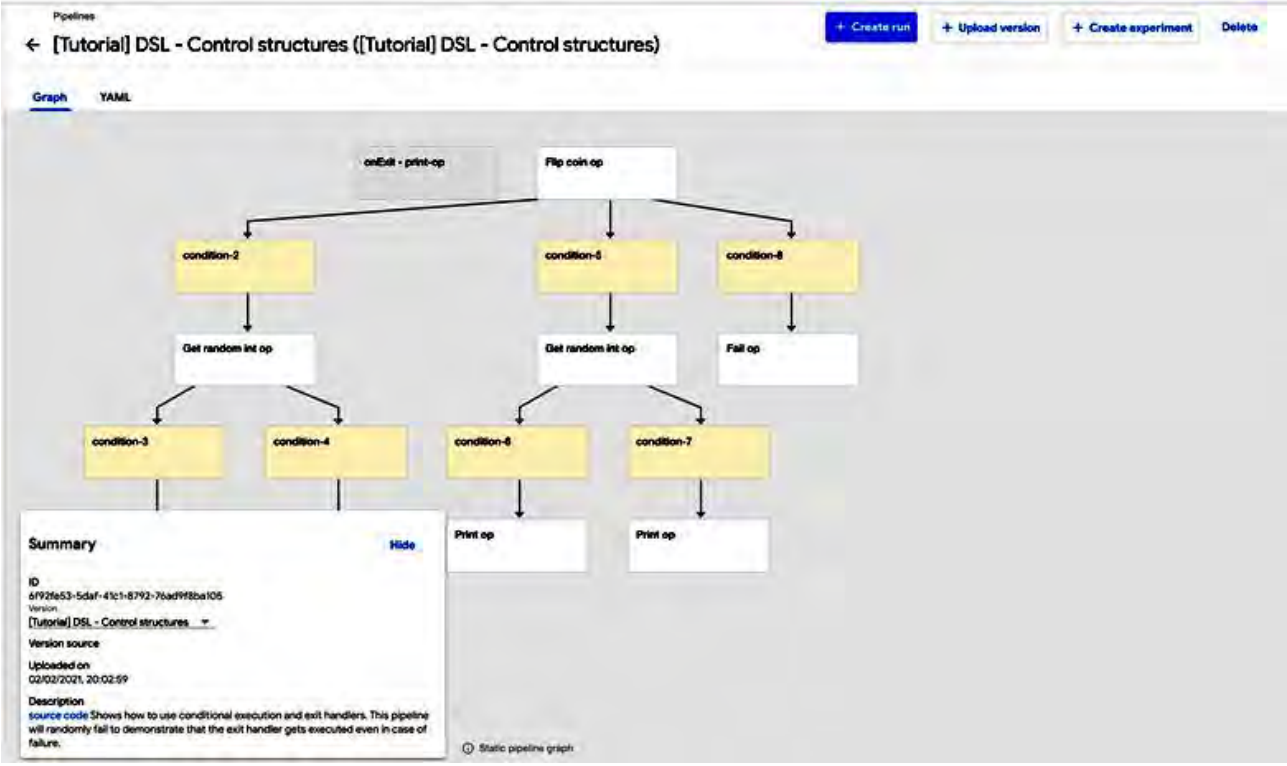


Рис. 2.8 Знімок екрана візуалізації Kubeflow pipeline

2.5 Моделювання у Kubeflow із застосування Seldon Core Serving

Сучасна методологія розробки програмних продуктів базується на принципах неперервного впровадження і не зупиняється коли модель потрапляє в продуктове середовище. Експлуатація моделі в класичному вигляді полягає в написанні коду, який би опрацьовував запити через grpc/http та повертав результат по відкритому порту. Також, при врахуванні технологічних рекомендацій, робота повинна включати створення контейнера, що в більшості дорівнює написанню Dockerfile, маніфести для розгортання в Kubernetes (deployment, service, ingress), розширення функціонала із відкриттям інтерфейсу по якому сторонні сервіси могли б отримувати метрики з роботи сервісу.

Для вирішення задачі експлуатації моделі в арсеналі Kubeflow є набір застосунків, таких як Seldon Core Serving, NVIDIA Triton Inference Server, BentoML, TensorFlow Serving.

Seldon Core це – open source платформа для експлуатації моделей МН в Kubernetes. Вона конвертує модель МН у REST/GRPC мікросервіс. Seldon Core підтримує розширення до тисячі екземплярів моделей і надає розширену підтримку із коробки такого функціонала, як метрики для моніторингу, логування запитів, пояснення моделі (explainers), виявлення крайніх випадків (outlier detectors), А/Б тестування, канареєчне розгортання. Здійснено практичний огляд Seldon Core Serving, хоча слід зазначити, що наведені застосунки мають схожий функціонал.

Для практичного аналізу було розгорнуто модель, яка перебуває у відкритому доступі, в Kubeflow кластер за допомогою маніфесту (рис. 2.9). Після розгортання маніфесту на виході було отримано наступний набір об’єктів які зображені на рис. 2.10 [9, 10].

```
! prepacked.yaml expirement/seldon-core/prepacked.yaml
1   ---
2   apiVersion: machinelearning.seldon.io/v1
3   kind: SeldonDeployment
4   metadata:
5     name: iris-model
6     namespace: seldon
7   spec:
8     name: iris
9     predictors:
10    - graph:
11        implementation: SKLEARN_SERVER
12        modelUri: gs://seldon-models/sklearn/iris
13        name: classifier
14      name: default
15      replicas: 1
16
```

Рис. 2.9 Маніфест розгортання SeldonDeployment

```
➜  ~ kubectl get all
NAME                                                  READY   STATUS    RESTARTS   AGE
pod/iris-model-default-0-classifier-b546f575b-hsgfn   2/2     Running   0          35h

NAME                                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
service/iris-model-default               ClusterIP   172.20.126.5    <none>        8000/TCP,5001/TCP   35h
service/iris-model-default-classifier    ClusterIP   172.20.95.216   <none>        9000/TCP            35h

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/iris-model-default-0-classifier   1/1     1            1           35h

NAME                                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/iris-model-default-0-classifier-b546f575b   1         1         1       35h
➜  ~ kubectl get virtualservices.networking.istio.io
NAME              GATEWAYS                       HOSTS   AGE
iris-model-grpc   [kubeflow/kubeflow-gateway]    [*]     35h
iris-model-http   [kubeflow/kubeflow-gateway]    [*]     35h
```

Рис. 2.10 Набір об’єктів, отриманих після розгортання маніфесту

Під парасолькою SeldenDeployment знаходиться сім окремих об’єктів Kubernetes. Такий рівень спрощення системи для кінцевого користувача здійснено з використання патерну операторів в Kubernetes. Оператор – це програмний застосунок, що використовує об’єкт custom resources для обслуговування інших застосунків та їх компонентів. В нашому випадку як оператор виступає pod-а seldon-controller-manager-5fc5dfc86c-lht2f, що є частиною Kubeflow кластеру.

Саме цей компонент разом із базовими Kuberentes контролерами відповідає за те, щоб при створенні custom resources seldondeployments.machinelearning.seldon.io, були створені об’єкти, наведені на рис. 2.10. Зокрема це: pod, service, deployment, replicaset та istio virtual service. Також, при видалені seldondeployments, всі підпорядковані компоненти будуть автоматично очищені, і кінцевому користувачу не обов’язково пам’ятати про вищеперелічені абстракції (service, deployment, pod тощо).

В Seldon core наявний розширений моніторинг з інтеграціями із Prometheus та Grafana, що можуть бути окремо встановленні із використанням пакетного менеджера helm. Метрики автоматично доступні на застосунки по шляху 8080/metrics і prometheus автоматично їх зчитує за допомогою механізму із назвою service autodiscovery (рис. 2.11) [9, 10].

Рис. 2.11 Знімок екрана моніторингової панелі Grafana на основі доступних метрик в Seldon core

2.6 Висновки до розділу

У розділі здійснено загальний огляд технології Kubernetes і розглянуто основні компоненти, необхідні для того, щоб мати повний та працюючий кластер Kubernetes. Kubernetes складається із набору контролерів, які регулярно стежать і узгоджують поточний стан додатків із заявленим цільовим станом. Досліджено оператори Kubernetes.

Досліджено контролери та їх похідні більш детально і те як вони можуть покращити існуючі програмні застосунки і вивести їх на новий рівень. Контролер є активним процесом перевірки стану, який відстежує фактичний та бажаний стан об’єктів. Він надсилає вказівки спробувати змінити поточний стан, щоб він був схожий на бажаний. Kubernetes використовує цей механізм зі своїми внутрішніми контролерами, тому можна повторно використовувати той самий механізм із власними контролерами.

Контролери можливі через хорошу модульність і управління подіями закладеними в архітектурі Kubernetes. Ця архітектура, природно, призводить до не монолітного і асинхронного підходу для контролерів як точок розширення.

Важливою перевагою тут є те, що отримується точна технічна межа між самим Kubernetes і будь-якими розширеннями. Однак одне питання асинхронного характеру контролерів полягає в тому, що їх часто важко відлагодити, оскільки потік подій не завжди простий. Як наслідок, не можна легко встановити точки відмови у контролері.

Також розглянуто як розширити платформу Kubernetes за допомогою таких контролерів, як оператори. Перш ніж користуватися оператором, слід уважно ознайомитись із конкретним випадком його використання, щоб визначити, чи відповідає він парадигмі Кубернетеса. У багатьох випадках звичайний контролер, який працює зі стандартними ресурсами, є досить хорошим. Такий підхід має ту перевагу, що він не потребує дозволу адміністратора кластера для реєстрації CRD, але має свої обмеження, коли мова йде про безпеку чи перевірку.

Оператор добре підходить для моделювання бізнес (користувацької) логіки домену, яка добре поєднується з декларативним способом обміну ресурсами з реактивними контролерами Kubernetes. Більш конкретно слід розглянути можливість використання оператора з CRD для застосовуваного домену для будь-якої з відомих ситуацій.

Якщо критерій бізнес використання відповідають встановленим, але необхідна більша гнучкість у здійсненні та збереженні ресурсів, доцільне використання власного сервера API. Якщо необхідний випадок використання не є декларативним, якщо дані для управління не вписуються в модель ресурсу Kubernetes або не потрібна тісна інтеграція в платформу, то, ймовірно, краще написати свій окремий API. Також необхідно ознайомитись додатково із документацією Kubernetes, яка містить розділ щодо пропозицій щодо використання контролера, оператора, користувацької реалізації API.

2.7 Список використаних джерел

1. The history of Kubernetes. November 2, 2023 by Stephanie Susnjara. [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/blog/kubernetes-history/ (дата звернення 01.11.2023).

2. Kubernetes: An Overview. Nov 7th, 2016 by Janakiram MSV. [Електронний ресурс] – Режим доступу до ресурсу: https://thenewstack.io/kubernetes-an-overview/ (дата звернення 01.11.2023).

3. The Kubernetes Effect. Feb. 07, 2018 by Bilgin Ibryam. [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoq.com/articles/kubernetes-effect/ (дата звернення 01.11.2023).

4. Cloud native compute foundation. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cncf.io/ (дата звернення 01.11.2023).

5. Сайфан Д. Осваиваем Kubernetes. Оркестрация контейнерных архитектур, 2016. – 400 с.

6. Michael H., Stefan S. Programming Kubernetes. Developing Cloud-Native Applications, 2019, 272 p.

7. B. Ibryam, R. Roland. Kubernetes Patterns: Reusable Elements for Designing Cloud-Native Applications, Sebastopol, United States of America. O'Reilly Media, Inc., 2019, pp. 150–202.

8. Red Hat Operators [Електронний ресурс] – Режим доступу до ресурсу: https://www.openshift.com/learn/topics/operators (дата звернення 01.11.2023).

9. M. Kuzmich, T. Gordiyenko, "Application of Kubeflow as a universal approach for the development and implementation of artificial intelligence systems", ARPN Journal of Engineering and Applied Sciences, vol. 18, 2023, no. 20, pp. 2311–2320 (Scopus).

10. Кузьміч М. Ю., Гордієнко Т. Б., "Застосування інструменту Kubeflow для інтеграції машинного навчання і штучного інтелекту в безпілотних літальних апаратах". Телекомунікаційні та інформаційні технології, 2023, вип. 3 (80), с. 65–78.

РОЗДІЛ 3
ВИКОРИСТАННЯ СИСТЕМИ РОЗПОДІЛЕНОЇ ОБРОБКИ ДАНИХ НА ОСНОВІ MESH-МЕРЕЖІ ТА ЇЇ ІНТЕГРАЦІЯ З ПОТОЧНОЮ ІНФРАСТРУКТУРОЮ

3.1 Основи побудови Mesh-мережі

Поняття Mesh (сітка) насамперед визначає принцип побудови мережі, відмінною рисою якої є архітектура, що самоорганізується, реалізуючи такі можливості:

створення зон суцільного інформаційного покриття великої площі;

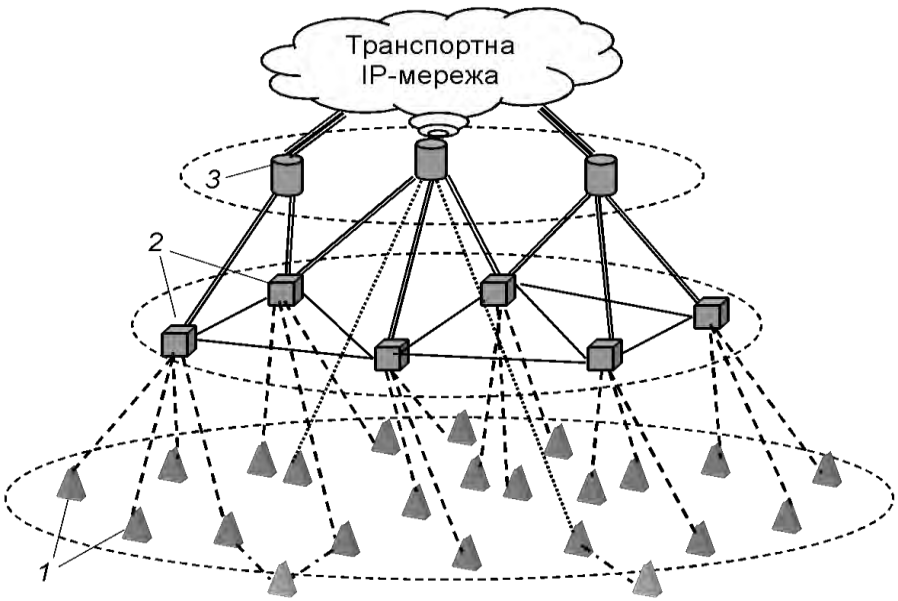
масштабованість мережі (збільшення площі зони покриття і щільності інформаційного забезпечення) у режимі самоорганізації;

використання безпровідних транспортних каналів (backhaul) для зв’язку точок доступу в режимі “кожний із кожним”;

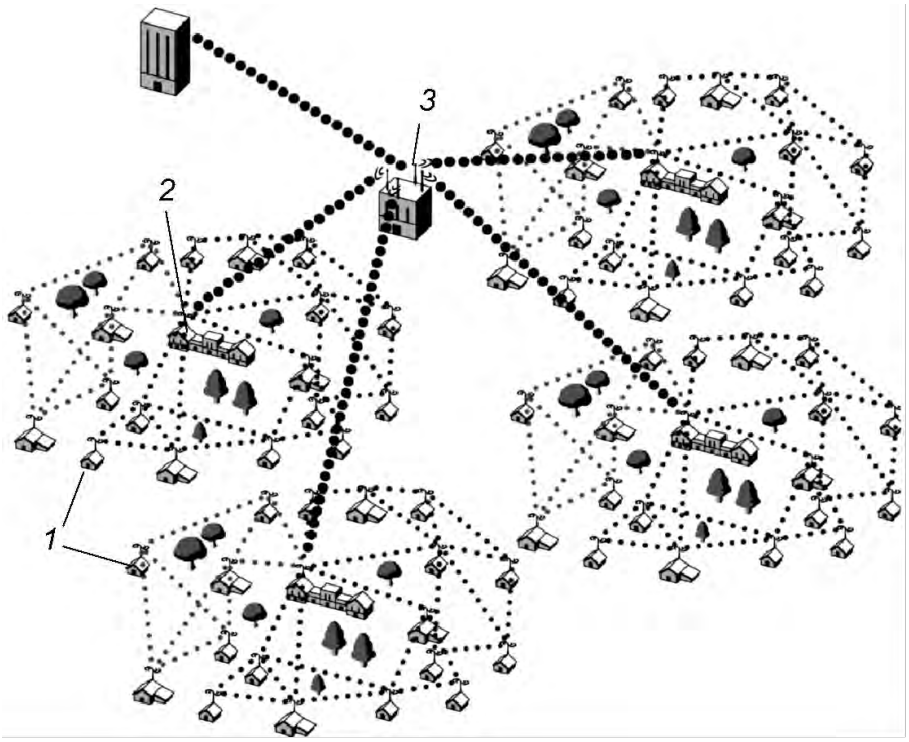
стійкість мережі до втрати окремих елементів.

Mesh-мережі будуються як сукупність кластерів (рис. 3.1) [1]. Територія покриття поділяється на кластерні зони, кількість яких теоретично не обмежене. В одному кластері розміщується від 8 до 16 точок доступу. Одна з таких точок є вузловою (gateway) і підключається до магістрального інформаційного каналу за допомогою оптичного кабелю чи по радіоканалу (з використанням систем широкосмугового доступу).

Вузлові точки доступу, так само як і абонентські точки доступу (Mesh-маршрутизатори) у кластері, з’єднуються між собою (з найближчими сусідами) по транспортному радіоканалу. Залежно від конкретного рішення точки доступу можуть виконувати функції ретранслятора (транспортний канал) або функції ретранслятора і абонентської точки доступу.



а



б

Рис. 3.1 Структура Mesh-мережі та її основні функціональні елементи
а – загальна структурна схема по рівнях доступу; б – мережа з однією вузловою точкою до-ступу; 1 – Mesh-клієнти (абоненти); 2 – Mesh-маршрутизатори (абонентські точки доступу); 3 – вузлові точки доступу (gateway)

Особливістю Mesh-мережі є використання спеціальних протоколів, що дають можливість кожній точці доступу створювати таблиці абонентів мережі з

контролем стану транспортного каналу і підтримкою динамічної маршрутизації трафіку по оптимальному маршруту між сусідніми точками. При відмові якої-небудь із них відбувається автоматичний перенаправлення трафіку по іншому маршруту, що гарантує не просто доставку трафіку адресатові, а доставку за мінімальний час. Процедура розширення мережі в межах кластера обмежується установкою нових точок доступу, інтеграція яких в існуючу мережу відбувається автоматично.

Недолік подібних мереж полягає в тому, що вони використовують проміжні пункти для передачі даних, а це може викликати затримку при пересиланні інформації і, як наслідок, знизити якість трафіку реального часу (наприклад, голосу чи відео). У зв'язку з цим існують обмеження на кількість точок доступу в одному кластері.

Таким чином, існує два типи безпровідних вузлів: перший – це Mesh-маршрутизатори, другий – Mesh-клієнти. Такий поділ зроблений з наступних мір-кувань: Mesh-маршрутизатор є вузлом з мінімальною рухливістю. Він виконує функції: агрегації навантаження безпровідної сіткової мережі WMN (Wireless Mesh Network) для передачі і прийому в зовнішні мережі; організує опорну мережу WMN; виконує функції маршрутизації внутрішнього навантаження. Для виконання таких завдань маршрутизатор оснащений необхідними інтерфейсами і програмним забезпеченням.

Такий підхід дає можливість максимально розвантажити Mesh-клієнтів відносно кількості інтерфейсів, а також програмного забезпечення за умови збереження основних постулатів WMN: самоорганізації й самонастроювання. Клієнти мережі відповідно до правил маршрутизації можуть встановлювати з'єднання як із своїми ж клієнтами, так і з клієнтами інших мереж, використовуючи маршрутизатор як шлюз.

У загальному випадку WMN можна розглядати як продовження розвитку мереж Ad hoc із урахуванням нових вимог до підвищення пропускної спроможності та надійності з'єднань у розподіленій безпровідній мережі. Основна відмінність WMN від Ad hoc виражається в існуванні в Mesh-мережі

конкретної інфраструктури. Завдяки стаціонарним (чи з мінімальною рухливістю) вузлам-маршрутизаторам у мережу вводяться імовірно розташовані вузли, навколо яких зосереджуються випадкові термінали. Однак WMN-мережа не виключає можливості з'єднання між собою випадкових терміналів без участі вузла-маршрутизатора.

Розглянемо Mesh-мережу за стандартом IEEE 802.16 [2], у якому Mesh-мережа розглядається як один із видів топології безпроводової мережі. Основна відмінність Mesh-мережі від архітектури "точка-багатоточка" полягає в тому, що якщо в останньому випадку точками А і С може спілкуватися тільки з точками Б і С, то в Mesh-мережі можлива взаємодія безпосередньо між точками А і С.

Оскільки мережі за стандартом IEEE 802.16 орієнтовані на роботу з широкими частотними каналами з високою пропускною спроможністю, то Mesh-мережі ввели до стандарту зовсім не для створення однорангових локальних мереж – для цього є стандарти серії IEEE 802.11 [3], які розробляє спеціальна робоча група Асоціації IEEE (рис. 3.2).

Причина в іншому – необхідно мати інструмент побудови широкосмугової мережі, в якій трафік може передаватися по ланцюжку з кількох станцій, ліквідовуючи тим самим проблеми передачі за відсутності прямої видимості. Відповідно і всі механізми керування, що в принципі дають можливість побудувати децентралізовану розподілену мережу, орієнтовані все ж таки на деревоподібну архітектуру з виділеною базовою станцією (основний вузол) і домінуючими потоками БС–АС.

У Mesh-мережі всі станції (вузли) формально рівноправні. Проте практично завжди обмін трафіку Mesh-мережі із зовнішнім оточенням відбувається через один певний вузол (рис. 3.3, 1 – абонентський термінал; 2 – точка доступу – базова станція). Такий вузол називають базовою станцією – БС чи точкою доступу) Mesh-мережі і саме на нього складають частину необхідних для керування Mesh-мережею функцій. При цьому керування доступом може відбуватися або на основі механізму розподіленого керування, або

централізованим способом – під керуванням БС. Також можлива і комбінація цих методів.

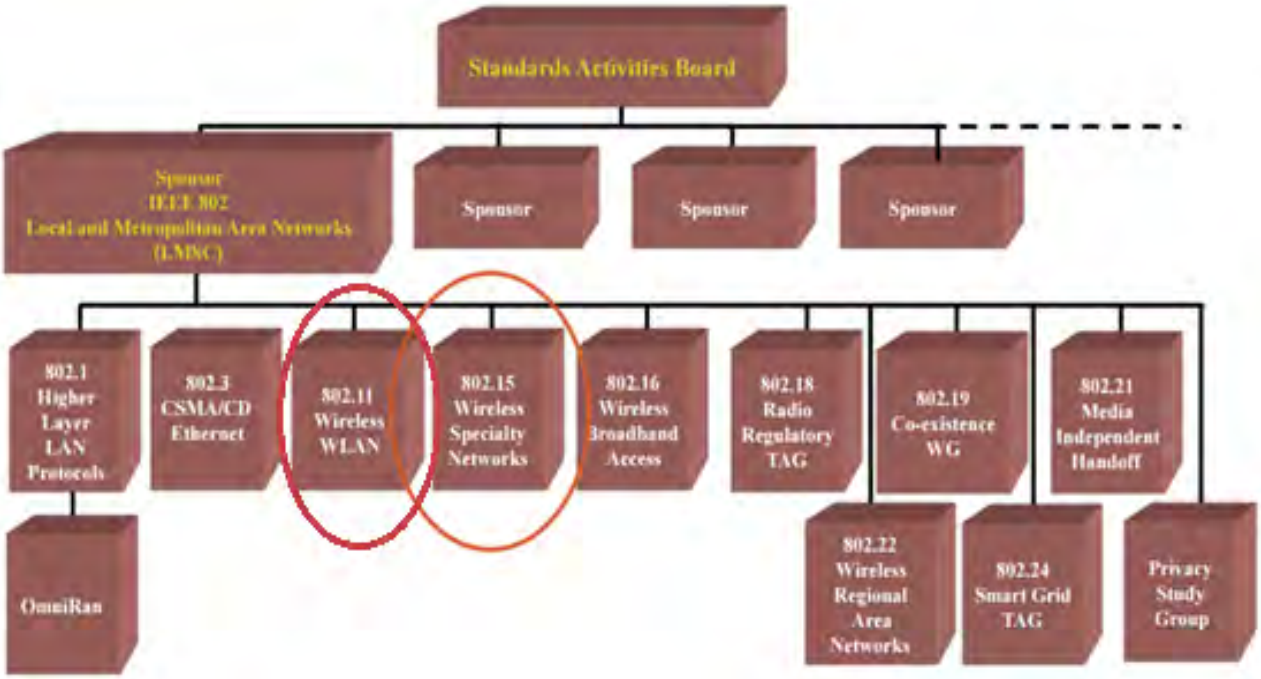


Рис. 3.2 Робочі групи Асоціації IEEE

Рис. 3.3 Приклад Mesh-мережі стандарту IEEE 802.16

Базове поняття в Mesh-мережі – сусіди, під чим розуміють всі вузли, які можуть встановлювати між собою безпосереднє з'єднання. Всі вони утворюють сусідське оточення. Вузли, пов'язані із заданим вузлом через сусідські вузли, називають сусідами другого порядку. Можуть бути сусіди третього порядку тощо.

У Mesh-мережі немає поняття висхідних/низхідних каналів. Весь обмін відбувається за допомогою кадрів. Станції передають повідомлення або у відведені їм часові інтервали (відповідно до попереднього призначення каналів), або отримують доступ до каналів довільним (випадковим) чином. Кожен вузол має унікальну 48-розрядну MAC-адресу (Medium Access Control address). Крім того, для ідентифікації всередині Mesh-мережі станціям присвоюється 16-розрядний мережевий ідентифікатор.

Кожен вузол постійно зберігає список даних про всіх своїх сусідів (відомості про віддаленість, сектори для направленої антени, показову необхідну потужність передавача для зв'язку, затримку поширення сигналу тощо) і транслює його в мережу із заданою періодичністю. На підставі цих списків від кожного з вузлів і відбувається керування мережею.

Кадр Mesh-мережі ділиться на керуючий субкадр і субкадр даних (рис. 3.4). Довжина керуючого субкадру – змінна величина, що задається БС. Керуючий субкадр становить набір пакетів MAC-рівня (Medium Access Control level) з тією різницею, що відразу після загального заголовку MAC-пакета (MAC pack) іде підзаголовок Mesh-мережі. Керуючий субкадр, залежно від реалізованих функцій, може бути двох типів – керування мережею (network control) і керування послідовністю доступу до каналів зв'язку (schedule control). У субкадрах завжди використовується модуляція QPSK (Quadrature Phase Shift Keying) із швидкістю кодування 1/2.

Рис. 3.4 Структура кадру Mesh-мережі

Субкадри керування вміщують інтервали для підключення до мережі нових пристроїв – «мережевий вхід» (network entry) і повідомлення «конфігурація мережі», які йдуть за ними. Повідомлення типу «конфігурація мережі» містять всю необхідну інформацію про склад мережі. Вони ж реалізують процедури керування. Ці повідомлення генерує кожен вузол і транслює по мережі через своє сусідське оточення. Серед інформації, яка передається, списки сусідів кожного вузла, ідентифікаційний номер БС і кількість її сусідів, номер логічного каналу для передачі трафіку доступу до каналів, віддаленість вузла (ранг сусідства) від БС тощо.

За допомогою таких повідомлень із заданою періодичністю транслюється дескриптор мережі – таблиця, що повністю описує поточні параметри мережі. Серед них – тривалість кадрів, довжина керуючого субкадру, кількість інтервалів

для повідомлень децентралізованого розподілу ресурсів, періодичність проходження субпакетів розподілу ресурсів, профілі пакетів, тип кодування, відповідність логічних каналів фізичним тощо. Дескриптор мережі передається від базової станції її сусідському оточенню, від нього – вузлам із наступним рангом сусідства тощо. Періодичність передачі дескриптора мережі нормована.

«Мережевий вхід» – це інтервал, протягом якого новий вузол може послати спеціальні повідомлення про свій намір підключитися до мережі (аналог інтервалу конкурентного доступу в мережі «точка-багатоточка»). Перед цим він повинен прийняти повідомлення про конфігурацію мережі, вибрати вузол для підключення, синхронізуватися з ним і лише потім відправляти запит. У відповідь вузол або відмовляє у доступі, або призначить новому вузлу мережевий ідентифікатор, канал і часовий інтервал для проведення процедур аутентифікації.

Розподіл канальних ресурсів у Mesh-мережі може бути централізованим і децентралізованим (розподіленим). У свою чергу децентралізований розподіл буває координованим з БС і не координованим з нею.

Децентралізований розподіл ресурсів передбачає, що розподіл відбувається в межах однієї групи сусідів (тобто між станціями, здатними безпосередньо зв’язуватися одна з одною). При координатному децентралізованому розподілі вузли обмінюються між собою спеціальними повідомленнями керування розподілом DSCH (Distributed Scheduling). Координованість полягає в тому, що період видачі таких повідомлень кожною станцією визначений і відомий її сусідам. DSCH-повідомлення – це запити на отримання канального ресурсу і повідомлення з відповіддю з наданням (підтвердженням) вільного ресурсу (часового інтервалу в субкадрі даних). Ресурс надається сусідом під конкретне з’єднання. Координовані DSCH-повідомлення передаються в субкадрах керування послідовністю доступу в обумовлених у мережному дескрипторі інтервалах. Некоординовані DSCH-повідомлення передаються в субкадрі даних.

Централізований розподіл ресурсів передбачає деревоподібну топологію мережі з БС у вершині. Воно реалізоване за допомогою двох типів повідомлень – централізованої конфігурації CSCF (Centralized Configuration) і централізованого планування CSCH (Centralized scheduling). Ці керуючі повідомлення розміщуються на початку субкадру керування трафіком доступу. Використовуючи повідомлення централізованого планування CSCH, кожен вузол визначає потребу в трафіку своїх дочірніх вузлів (тобто трафік яких від (до) БС проходить через певний вузол) і повідомляє свою потребу вузлу, що знаходиться вище, – аж до БС. Про-аналізувавши потребу, БС розсилає повідомлення CSCH, інформуючи кожен вузол про виділену йому смугу пропускання (у біт/с) у висхідному і низхідному напрямку. Виходячи з цих даних, кожен вузол вже сам «запрошує» (або призначає) пакети у субкадр даних у (для) своїх сусідських вузлів за допомогою повідомлень децентралізованого планування DSCH.

Повідомлення централізованої конфігурації CSCF формуються БС і транслюються по мережі для інформування всіх її вузлів про поточний стан. CSCF вводить таку інформацію, як кількість доступних логічних каналів і їх перелік, перелік вузлів у мережі з вказівкою кількості дочірніх вузлів для кожного з них, а також профілі висхідних/низхідних пакетів для кожного дочірнього вузла.

3.2 Mesh-мережа за стандартом IEEE 802.11s

3.2.1 Загальні характеристики Mesh-мережі

Однією з технологій, яка може бути впроваджена протягом найближчих років є бездротовий Mesh-мережі WMN. Бездротові Mesh-технології були відомі майже так само довго, як бездротові локальні мереж WLAN (Wireless Local Area Network), але вони ніколи ще не мали такої популярності в бездротовій індустрії, як останнім часом. Оскільки популярність постійно росте, кінцеві користувачі вимагають високу пропускну здатність, більш широке охоплення, а також підвищену надійність. На сьогоднішній день не існує офіційного стандарту для

WNM. Однак, стандари з'являтися з вимогами ринку й інтересу та фінансових інвестицій до галузі. Індустрій представляються як єдине ціле в різних робочих групах по стандартизації IEEE 802. Ці стандарти можуть стимулювати повсюдне розгортання WMN по всьому світі [4, 5].

Кілька нових і комерційно цікавих додатків для високошвидкісних WLAN Mesh нещодавно була розгорнуті. Технології WLAN Mesh є економічно життєздатною мережевою парадигмою для побудови широкосмугових, муніципальних, громадської безпеки мереж, тактичних, і також телекомунікаційне рішення для так званих гарячих зон. Оригінальна архітектура mesh мережі яка розвинулась з мобільних Ad hoc мереж MANET (Mobile Ad-hoc Networks) використовується для військових цілей [6].

Спеціальна робоча група IEEE розробляє різні протоколи для MANET протягом майже десяти років. Mesh-мережі відрізняються від MANET через більш структуровану форму взаємозв'язку, і не прямі, однорангові комунікації. Це призводить до природної ієрархії в мережі. Mesh-мережа стала популярною, і багато виробників почав виготовляти продукцію для Mesh-мереж, потреба в стандартизації і взаємодії стала очевидною з 2003 року.

Стандарт IEEE 802.11s [7] був прийнятий з наміром розширити мережу WLAN для EES Mesh-мереж. Існуючі стандарти IEEE 802.11 визначали мережевий доступ до операцій між клієнтами бездротової локальної мережі та точками доступу. Для того, щоб поширити стандарт IEEE 802.11 для Mesh-мереж, в транспортні (інфраструктури WLAN і LAN – Local Area Network) і шлюзові (інфраструктури підключення до дротової локальної мережі) операції повинні були бути внесені зміни (рис. 3.5).

Зазначені операції розповсюджувались на управління доступом до середовища передачі MAC, енергозбереження, маршрутизацію, взаємодію з мережами на базі стандартів серії IEEE 802, безпеки, якості обслуговування, управління і налаштування мережі WLAN Mesh. Поточний базовий проект був створений шляхом об'єднанням пропозицій від двох підключених промислових груп SEE-Mesh і WiMesh, що складались з більш ніж 25 компаній та

університетів. Пропозиції надходили з таких питань як дослідження, ціноутворення постачальників і розробників систем, інтегрування та розповсюдження технології тощо. Необхідно виконати ще достатньо робіт для остаточного успішного завершення впровадження цього стандарту і виведення цієї технології на максимальну продуктивність.

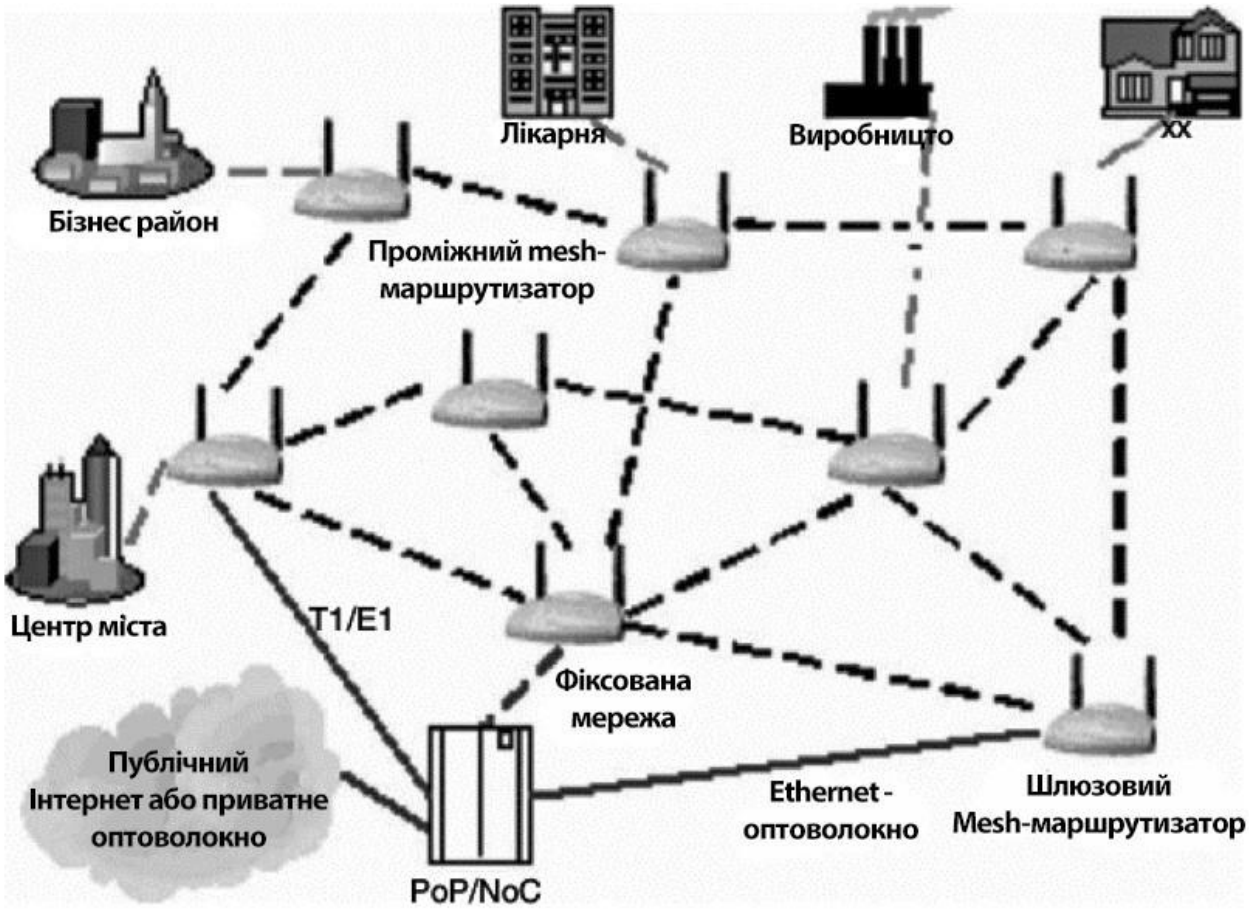


Рис. 3.5 Бездротова LAN Mesh мережа

3.2.2 Підключення WLAN Mesh

Бездротова мережа WLAN, що працює з ретрансляторами зв’язку для пересилання трафіку на шляху від провідної точки входу в Інтернет, повинна відповідати стандартам серії IEEE 802.11. Розвиток вимог за стандартами IEEE 802.11a (1999 рік) і IEEE 802.11g (2003 рік) [8] дозволив збільшити швидкість передачі даних в мережах WLAN за допомогою спектрально-ефективних видів модуляції до 54 Мбіт/с. За стандартом IEEE 802.11n (2009 рік) на основі

технології множинного вводу-виводу швидкість передачі даних збільшена до 600 Мбіт/с, а за стандартом IEEE 802.11ay (2019 рік) – до 10 Гбіт/с [9].

WLAN Mesh використовує пристрої фізичного і канального рівня на основі стандарту IEEE 802.11 для забезпечення функціональності мережі ESS (англ. Employee Self Service). Точки доступу за стандартом IEEE 802.11 MP (Mesh Point) при використанні в мережі WLAN, встановлюють бездротове з'єднання один з одним, для активування автоматичної топології навчання і динамічної конфігурації шляху. MP-to-MP зв'язок утворює магістральну мережу, яка надає користувачам з низькою вартістю, з високою пропускною здатністю, і безшовні ретрансляторними з'єднання з обмеженим числом точок входу в Інтернет і зв'язок з іншими користувачами у цій Mesh-мережі. Кожен MP при бажанні може надати бездротове підключення до користувачів, відома як Mesh доступ. Ці пристрої називаються точками доступу MAP (Mesh Access Point).

На рис. 3.6 показана типова мережа WLAN Mesh та її різних компонентів (AP – точками доступу, Access Point; STA – абонентські термінальні станції, Station Abonent Terminal).

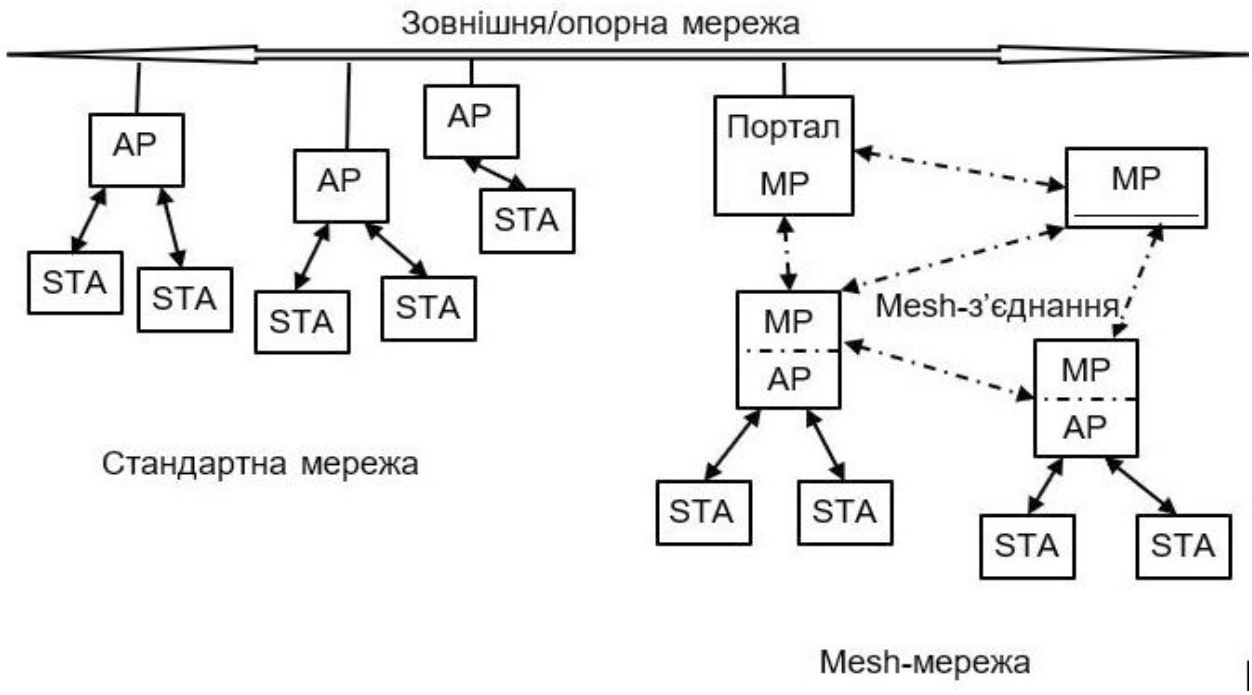


Рис. 3.6 Архітектура мережі за стандартом IEEE 802.11 і її Mesh конфігурації

Бездротова локальна Mesh-мережа визначається як WLAN Mesh на основі стандарту IEEE 802.11 бездротової розподільчою системи WDS (Wireless Distribution System), яка є частиною DS (Distribution System), що складається з набору двох або більше точок доступу Mesh взаємопов'язаних за стандартом IEEE 802.11, що посилаються і обмінюються даними через WLAN Mesh сервіси. Мережа WLAN Mesh може підтримувати нуль або більше точок входу, систему автоматичного навчання топології і динамічний вибору маршруту.

Стандарт IEEE 802.11s [7] описує можливість побудови Mesh-мережі на основі модернізованих технологій бездротового вірність відтворення (Wireless Fidelity). В існуючих мережах стандарту 802.11 термінальні абонентські станції STA пов'язані з точками доступу AP і можуть взаємодіяти тільки з ними. AP мають вихід в інші мережі (наприклад, Ethernet), але не можуть обмінюватися інформацією один з одним (рис. 3.6). У Mesh-мережі, крім термінальних станцій і точок доступу, присутні особливі пристрої – вузли Mesh-мережі MP, здатні взаємодіяти один з одним і підтримують Mesh-служби (рис. 3.5).

Один пристрій може поєднувати кілька функцій. Так, MP, суміщені з точками доступу, називаються точками доступу Mesh-мережі MAP. Портали Mesh-мережі MPP (Mesh Point Portal), будучи MP, з'єднують Mesh-мережу із зовнішніми мережами. Таким чином, Mesh-мережу з точки зору інших пристроїв і протоколів більш високого рівня функціонально еквівалентна широкомовній Ethernet-мережі, всі вузли якої безпосередньо з'єднані на канальному рівні.

Зміни, наведені у в стандарті IEEE 802.11s, практично не зачіпали фізичний рівень. Всі нововведення відносяться до MAC-підрівню канального рівня. Крім того, в цьому стандарті розглядаються питання маршрутизації пакетів в рамках Mesh-мережі (фактично – мережевий і транспортний рівень моделі OSI), що виходить за початкові рамки стандарту IEEE 802.11a.

Структура пакетів MAC-рівня в Mesh-мережі аналогічна стандартному формату пакетів мереж за стандартом IEEE 802.11. Формат заголовка MAC-пакета в Mesh-мережі така ж, як і в стандарті IEEE 802.11 (за винятком поля контролю високої пропускної здатності HTC (High Throughput Control),

призначеного, для підтримки обладнання стандарту IEEE 802.11n. Перші три поля заголовка і поле контрольної суми послідовності перевірки кадрів FCS (Frame Check Sequence) присутні у всіх пакетах MAC-рівня. MAC-пакет за стандартом IEEE 802.11s відрізняється Mesh-заголовком на початку поля даних. Він присутній в пакетах даних тоді і тільки тоді, коли вони передаються від Mesh-вузла до Mesh-вузла за встановленим з'єднанням, а також приєднується до керуючих пакетів типу багатоперехідної дії (multihop action).

Наявний детермінований доступ в Mesh-мережі MDA (Mesh Deterministic Access) забезпечує опціональний механізм, який дозволяє отримувати доступ до середовища в заздалегідь зарезервовані часові інтервали. Це знижує конкуренцію доступу до середовища передачі, що істотно збільшує ймовірність своєчасної доставки даних, чутливих до затримок (аудіо і відеопотоки, дані з високим пріоритетом тощо).

MDA-з'єднання можливе тільки між станціями, що підтримують даний механізм. MDA-резервування задає інтервали, протягом яких MDA-станції не намагаються передавати пакети, щоб не заважати передачі даних по зарезервованому каналу. Створення MDA-з'єднання ініціюється вузлом-джерелом даних, а приймається або відхиляється вузлом-адресатом даних. Для встановлення MDA-з'єднання пристрій вибирає інтервали часу, не зайняті іншими MDA-з'єднаннями, про які йому відомо. Якщо досягнута межа допустимої кількості MDA-з'єднань для сусідів певного вузла, то пристрій не створює нове MDA-з'єднання.

Сумісність пристроїв від різних виробників в одній мережі забезпечує концепція профілів. Профіль містить власне ідентифікатор профілю, ідентифікатор протоколу маршрутизації і ідентифікатор метрики протоколу маршрутизації. Пристрій здатний підтримувати декілька профілів роботи, але лише один з них може бути активним. Обов'язковий для всіх пристроїв за стандартом IEEE 802.11s профіль використовує гібридний безпровідний Mesh-протокол маршрутизації HWMP (Hybrid Wireless Mesh Protocol) і метрику часу передачі в каналі ALM (Airtime Link Metric).

Механізм установки з’єднань заснований на періодичній посилці стандартно-го повідомлення «відкрити з’єднання». У відповідь на нього може бути отримано повідомлення «підтвердження з’єднання» або «закриття з’єднання». З’єднання між двома сусідніми МР вважається встановленим тоді і тільки тоді, коли обидва МР послали один одному команди «відкрити з’єднання» і відповіли підтвердженням з’єднання. Для кожного встановленого з’єднання передбачено час життя, протягом якого воно повинно бути використано або підтверджено.

3.3 Пропускна здатність і менеджмент ресурсів у Mesh-мережі

Традиційні Ad hoc мережі та WMN базуються на однокальнольному радіо інтерфейсі. WMN, незалежно від її простоти і високої відмовостійкості, зіткнулося з істотним обмеженням обмеженням пропускної здатність мережі. У той час як теоретичний верхня межа пропускної здатності вузла асимптотично обмежена виразом

$$\mathrm{Q}\left(1/\sqrt{n}\right), \tag{3.1}$$

де n – це число вузлів в мережі,

теоретично досяжною пропускною здатністю кожного вузла у випадковій ad hoc мережі з ідеальним глобальним плануванням і вибором маршрутизації визначається за виразом

$$\mathrm{Q}\left[\left(1/\sqrt{n}\right)\log n\right]. \tag{3.2}$$

Таким чином, зі збільшенням числа вузлів в мережі, пропускна здатність стає неприйнятно низькою. При використанні реального MAC, маршрутизації та транспортних протоколів і реалістичної моделі трафіку, досяжна пропускна в WMN, на практиці, набагато менше, ніж теоретичний верхня межа. В цілому, пропускна здатність, яка досяжна в довільній WMN пропорційна виразу

$$\mathrm{Q}\left[W \cdot n^{-1/d}\right], \tag{3.3}$$

де d – розмірність мережі; W – загальна пропускна здатність.

Для двомірних мереж, пропускна здатність може визначатись за виразом Q Q$\left[W \cdot n^{-1/2}\right]$. Один з підходів до поліпшення пропускної спроможності WMN є використання декількох радіоінтерфейсів. Хоча верхня межа пропускної здатності не залежить від того, як полоса пропускання буда поділена між декількома інтерфейсами, на практиці, з реалістичними MAC і протоколи маршрутизації, пропускна здатність може бути значно збільшена за рахунок використання декількох інтерфейсів і тонкої настройки протоколів маршрутизації. Хоча останнім часом розвиток WMN з використанням декількох радіоінтерфейсів досяг важливих успіхів з появою недорогих радіоінтерфейсів, які підтримують стандарт IEEE 802.11. У той час як MR-WMN (Multi Radio WMN) запропонували кілька переваг, таких як збільшення пропускної здатності мережі, вони також стикаються з низкою запитань і проблем.

Пропускна здатність обмежена для WMN-вузлів в одноканальній системі в порівнянні з багатоканальної системою. Вона швидко деградує в міру збільшення довжини шляху. Також наявні фактори, що підсилюють деградація пропускної здатності, такі як характеристики протоколу MAC, проблема відкритого вузла, прихована проблема з термінальним устаткуванням та збільшення значення відношення бітових помилок BER (Bit Error Rate), посилюються в одноканальній системі. Наприклад, як показано на рис. 3.7, коли вузол 1 передає на вузол 2, особливо коли використовуються MAC протоколи на основі CSMA/CA, вузли 2 і 3 не можуть ініціювати обмін інформації, оскільки вузол 2 резервується для передачі інформації для вузла 1. Це є проблемою відкритого вузла в WMN.

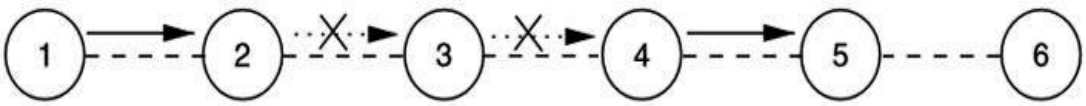


Рис. 3.7 Приклад лінійної топології

Експериментальним шляхом визначено, що зменшення пропускної спроможності для одноканальної мережі прямопропорційно кількістю хопів.

$$Q(1/n), \tag{3.4}$$

де n – кількість хопів.

Коли кількість хопів 5 або більше, значення пропускної дуже мале. Хоча існують і інші чинники, такі як природа протоколу маршрутизації, «жадібність» початкових вузлів, що спричиняє «голод» останніх в ланцюзі, поведінка протоколів MAC тощо. Отже одними з основних проблем, з якими стикаються WMN є такі як пропускна здатність та масштабованість мережі.

Менеджмент ресурсів у Mesh-мережі включає в себе ефективне управління мережевими ресурсами, такими як енергетика, пропускна здатність, інтерфейси і місця для зберігання інформації. Наприклад, енергетичні ресурси можуть бути ефективно використані в WMN, якщо кожен вузол в системі має новий інтерфейс з низьким рівнем потужності на додаток до звичайного інтерфейсу, оскільки загальне споживання потужності, навіть в режимі очікування, дуже сильно залежить від типу інтерфейсу.

Полоса пропускання також може бути використана більш ефективно в середовищі багатоканальної передачі. Балансування навантаження між декількома інтерфейсами може допомогти запобігти утворення ефекту «горла пляшки» для конкретного каналу. Пропускна здатність кожного інтерфейсу можуть бути об’єднанні для отримання високої швидкості передачі даних. В такому механізмі агрегації пропускної здатності (також відомий, як чергування смуги пропускання), використання протоколу динамічного планування передачі пакета може бути використано, щоб отримати більш високу продуктивність. Одним з важливих переваг використання багатоканальної системи в WMN є можливість надання користувачу необхідної якості обслуговування QoS (Quality of Service) для його різноманітних потреб шляхом диференціації сервісів.

3.4 Проблемні питання впровадження Mesh-мереж

3.4.1 Особливості мережевих архітектур

Існує низка питань, які необхідно враховувати при практичному використанні WMN. Питання проектування можна поділити на архітектурні питання і проблеми протоколу. WMN може бути розроблена в трьох різних мережевих архітектурах, заснованих на топології мережі: плоска WMN; ієрархічна WMN; гібридна WMN.

В плоских WMN, мережа формується на клієнтських машинах, які діють одночасно як хост (користувач), так і маршрутизатор. Тут кожен вузол знаходиться на тому ж рівні, що і його аналоги. Вузли бездротових клієнтів координуються між собою для забезпечення маршрутизації, конфігурації мережі, надання різноманітних сервісів та послуг, а також інші додатки ініціалізації. Ця архітектура найближче до Ad hoc мережі, тому це найпростіший випадок серед всіх WMN архітектур. Основна перевага цієї архітектури полягає в її простоті, а до її недоліків можна віднести відсутність масштабованості мережі. Основні проблеми при проектуванні плоскої WMN є схеми адресації, маршрутизація і схеми розпізнавання служб.

В ієрархічній WMN, мережа має кілька ярусів або ієрархічних рівнів, в яких WMN клієнтський вузол найнижчий в ієрархії. Ці клієнтські вузли можуть взаємодіяти з магістральної мережі, утвореної маршрутизаторами. В більшістю випадках, звичайні вузли за необхідності можуть виконувати функції спеціалізованих нод, утворюючи магістральну WMN мережу. Магістральні вузли не можуть створювати або опрацювувати трафік даних як клієнтські вузли. Самоорганізація і підтримка магістральної мережі забезпечується за рахунок WMN маршрутизаторів, деякі з яких в магістральній мережі можуть мати інтерфейси доступу до мережі Інтернет і такі вузли називаються шлюзовими вузлами.

Гібридна WMN – це особливий випадок ієрархічних WMN, яка використовує інші WMN для передачі даних. Наприклад, використовуючи різноманітні Mesh інфраструктури таких як стільникові мережі, WiLL мережі,

WiMAX або супутниковий зв’язок. Приклади таких гібридних WMN включають багатоінтервальні стільникові мережі [10], пропускна, якої збільшена за рахунок використання бездротових місцевих петлевих (TWiLL) мереж, і об’єднанні їх з стільниковою мережею [11]. Практичним застосуванням таких гібридних WMN можна назвати CalMesh платформу [12], що ефективна та використовується під час надзвичайних ситуацій. Ці гібридні WMN можуть використовувати різні типи технологій для транспортної і магістральні Mesh мережі. Так як розвиток цієї архітектури дуже залежить від того, як вона працює з іншими існуючими бездротовими мережевими рішеннями, то вона стає дуже важливою віткою у розвитку Mesh-мереж.

3.4.2. Проблеми за рівнями взаємозв’язку відкритих систем

Питання розробки протоколів взаємозв’язку відкритих систем OSI (Open Systems Interconnection) можуть бути описані пошарово починаючи від фізичного до прикладного рівнів. На фізичному рівні основним завданням є вибір відповідної технології радіозв’язку. Вибір технології радіозв’язку можуть бути засновані виходячи від технологічних і економічних міркувань. Основні технологічні міркування включають: спектральну ефективність; швидкість передачі даних на фізичному рівні; завадостійкість системи.

Наприклад, вибір технологій, таких як множинний доступ з кодовим поділом каналів CDMA (Code Division Multiple Access), використання надширокої смуги частот UWB (Ultra-Wide Band), а також технології MIMO (Multiple Input Multiple Output) є більш придатними для WMN фізичного рівня, ніж найбільш популярною технології фізичного рівня, з ортогональним частотним розділенням каналів мультиплексування OFDM (Orthogonal Frequency-Division Multiple Access), який використовується в сучасному WMN. Наприклад, сучасні технології фізичного рівня, в першу чергу на основі OFDM забезпечують максимальну швидкість передачі даних фізичного рівня до 54 Мбіт.

У мережі з високою щільністю та наявністю різноманітних перешкод, ця значення можу бути незадовільним. Таким чином, розробка нових і високошвидкісних методів передачі даних фізичного рівня, таких як UWB є основним завданням на фізичному рівні. На додаток до вибору конкретної технології фізичного рівня, програмовані радіоприймачі або когнітивне радіо додає ще один вимір до конструкцій на фізичному рівні. Ця потреба породжена вимогами при надзвичайних ситуаціях або військових спецоперацій, де спектр, який використовується для зв'язку залежить від невикористаного частотного спектру в певній місцевості. У таких додатках, програмне забезпечення певного радіо з когнітивними можливостями буде ідеальним вибором.

Крім технологічних міркувань, другою найбільш важливою вимогою є економічний або соціальні фактори, де простота технології фізичного рівня призведе до недорогих пристроїв і, як наслідок, більша кількість людей матимуть доступ до використання даної технології. Прикладом цього є успіхи стандарту IEEE 802.11b, на основі якого розроблені недорогі мережеві карти, що внесли свій вклад у поширення WMN. При виборі технології фізичного рівня, мережевий інженер повинен приділяти увагу на додатки і сценарії, що знаходяться на стороні користувача.

Конструкція протоколу канального рівня займає ключове значення, оскільки максимальна пропускна спроможність в значній мірі залежить від продуктивності протоколів MAC. У доповненні до повністю розподілених операцій, основні проблеми, з якими стикаються при реалізації стандарту IEEE 802.11, що вивчає технологію на базі CSMA/CA (Carrier Sense Multiple Access With Collision Avoidance) з функцією розподіленої координації DCF (Distributed Coordination Function), є: приховані термінальні проблемами; проблеми захищеності інформації терміналу; визначення місцезнаходження; висока ймовірність помилки в каналі.

Для того, щоб підвищити пропускну здатність мережі, використовується безліч радіостанцій, які працюють в декількох каналах. Таким чином, нові протоколи MAC повинні бути призначені для роботи в багатоканальних

системах. Протоколи канального рівня також повинні бути пристосовані для роботи в різних технологіях фізичного рівня, такі як UWB і MIMO. Ще один популярним питанням дослідження для підвищення продуктивності MAC є використанням механізмів кросрівневої взаємодії, які дозволяють протоколам MAC використовувати інформацію з інших рівнів. У традиційних бездротових або дротових мережах, кожен шар працює зі своєю власною інформацією. В цілому, конструкція протоколу MAC-рівня повинна включати в себе методи і рішення, щоб забезпечити кращу масштабованість мережі і пропускну здатність.

На відміну від протоколів маршрутизації (мережевий рівень) для Ad hoc мереж, протоколи маршрутизації, в залежності від мережевого сценарію, стикаються з різними завданнями в розробці. Оскільки WMN відносно статична мережа, маршрутизація може використовувати табличні підходи маршрутизації, такі як в провідних мережах або е Ad hoc мережах [13]. Основні проблеми, з якими стикаються протоколи маршрутизації в WMN є: конструкція метрики маршрутизації; накладні витрати мінімальної маршрутизації; живучість маршруту; ефективне використання інфраструктури підтримки; балансування навантаження; адаптивність маршрут.

Розробка метрики маршрутизації відіграє вирішальну роль в досягненні високої продуктивності. Краща метрика маршрутизації може також відрізнятися за своїми характеристиками. В WMN маршрутна метрика конструкція повинна приймати якість сигналу канального рівня до уваги для кращої продуктивності. Протоколи маршрутизації для WMN, забезпечуючи при цьому хорошу продуктивність, також повинні споживати мінімальну частину пропускної здатності для створення шляху.

Крім того, використання бездротового середовища вимагає можливості швидкого шляху реконфігурації, щоб зберегти стійкість шляху. Іншим важливим аспектом є балансування навантаження, яке повинно бути включеним до протоколу маршрутизації. Нарешті, протокол маршрутизації для WMN повинен бути максимально адаптований до конкретної динамічної мережі. Протокол маршрутизації може бути класифікований в будь-який плоский протокол

маршрутизації або ієрархічний протокол маршрутизації на основі типу мережі, де застосовуються протокол маршрутизації.

Основним завданням на транспортному рівні є ефективність транспортних протоколів в WMN. Оскільки для WMN характерне велике значення часу проходження маршруту RTT (Route-Trip-Time), варіації зміни RTT залежать від кількості хопів на шляху. Втрата пакетів, колізії, асиметричність архітектури мережі, і помилки в з'єднанні також негативно впливають на продуктивність.

Популярний протокол транспортного рівня TCP (Transmission Control Protocol) в мережах Internet дуже неефективний в WMN. Транспортний рівень повинен бути уточнений або переписаний для підвищення його ефективності на WMN. Деякі завдання, які постають при конструюванні протоколів транспортного рівня для WMN: надійність від кінця до кінця; пропускна здатність; можливості для обробки мережевої асиметрії; можливість для обробки динамічності мережі.

Найбільш популярний сервіс (рівень додатків) для WMN це послуга доступу в Інтернет. По суті, WMN повинен надавати інтернет-послуги для житлових районів і підприємств. Також дуже важливо забезпечити підтримку чутливих до часу видів трафіку, де великі затримки є неприйнятними. Оскільки більшість мережевих послуг знаходяться в повністю розподіленій формі, статичні механізми виявлення конкретного сервісу не можуть бути ефективними в WMN. Іншою важливою вимогою при проектуванні протоколу прикладного рівня є робота з гетерогенністю мережі, оскільки дані можуть проходити через різні мережі перед доставкою до кінцевого застосування.

3.4.3 Питання безпеки в бездротових Mesh-мережах

WMN схильна тим же основним спільним загрозам як для провідних, так і безпровідних мереж: повідомлення можуть бути перехоплені, змінені, затримані на певний час, повторно відтворені, або нові повідомлення можуть бути вставлені на місце старих. Може мережу і надає ресурси доступні без авторизації, і вони можуть бути недоступні на відмову в обслуговуванні атак.

Універсальні служби безпеки, боротьбі з цими загрозами є такі:

1) *конфіденційність* – відомості, що відкрито тільки для цільової аудиторії;

2) *цілісність* – дані не можуть бути змінені без попередження;

3) *автентифікація* – підприємство має насправді особистість претензій до нього не мають;

4) *контроль доступу* гарантує, що тільки авторизовані дії можуть бути виконані;

5) *непідроність* – захищає від суб'єктів, що беруть участь в комунікації, які можуть пізніше брехливо заперечувати, що обмін не відбувся;

6) *доступність* – гарантує, що авторизаційні дії можуть насправді мати місце.

Подальші служби безпеки базуються на забезпеченні конфіденційності користувачів (анонімність, трасування і профілювання користувача), транспортному потоку конфіденційності (суб'єкти спілкування), або навіть захисту того, що спілкування взагалі відбулося.

Захист зв'язку передбачає забезпечення конфіденційності (шифрування), аутентифікація партнерів по спілкуванню, а також захист цілісності та автентичності повідомлень. Захист цілісності відноситься не тільки до цілісності окремого повідомлення, але і правильного порядку повідомлень, аналізуючи зміни або видалення в даних. Ці технології також можуть бути використані в межах mesh-мережі для автентифікації Mesh-вузлів і встановлення сеансових ключів, які захищають конфіденційність і цілісність трафіку обміну даних між ними.

Трафік може бути захищений на різних рівнях (канальний рівень, мережевий рівень, транспортний рівень, з допомогою спеціального програмного забезпечення), особливо в системах бездротового зв'язку в стандарту(GSM, UMTS, DECT, IEEE 802.11 WLAN і Bluetooth, WiMAX 802.16), які включають в себе засоби для захисту бездротового з'єднання. Вони використовують різні схеми інкапсуляції кадрів, різних протоколів аутентифікації, і різних алгоритмів шифрування.

Бездротова локальна мережа WLAN на основі стандарту IEEE 802.11i [14] (в мережі Wi-Fi захищеного доступу: шифрування WPA – Wi-Fi Protected Access і WPA2) підтримує два режими безпеки: загальний ключ налаштований на підключення пристроїв, який часто використовується в домашніх мережах, або користувачі можуть пройти аутентифікацію на сервері автентифікації (сервера AAA – Authentication, Authorization, Accounting). Для цієї мети, використовується розширений протокол аутентифікації EAP (Extensible Authentication Protocol). Фактичну процес аутентифікації проходить між мобільною станцією (MC) і AAA-сервером за допомогою протоколу EAP (рис. 3.8).

Рис. 3.8 Доступ до WLAN на основі протоколу EAP

EAP, виступає транспортом між МС і точкою доступу AP, використовуючи протокол або розширюваний протокол автентифікації через локальну мережу EAPOL (Protocol or Extensible Authentication Protocol over LAN), і між точкою доступу і сервером AAA по протоколу Radius. У разі успіху, майстер-ключ сеансу видається, що передається від сервера аутентифікації AS (Authentication Server) до точки доступу бездротової мережі. Він використовується як вхідні дані для підключення 4-смугового протоколу авторизації, який встановлює тимчасовий сеансовий ключ для захисту бездротового зв'язку. Цей ключ використовується для захисту користувача трафіку, використовуючи протокол цілісності тимчасового ключа TKIP (Temporal Key Integrity Protocol).

Існують різні методи EAP для автентифікації на основі цифрових сертифікатів (EAP-TLS), паролів (EAP-MSChapV2), або повторне використання мобільних протоколів мережевої аутентифікації (EAP-SIM, EAP-AKA). WLAN EAP-доступ використовується зокрема для корпоративних мереж і публічних хот-спотів, де база даних користувачів є доступною.

Транспортні комунікації також можуть бути захищені вище канального рівня. IPSec (скорочення від англ. IP Security) захищає IP-трафік в мережі (рівень IP – Internet Protocol). Архітектура IPSec задає два протоколи безпеки: інкапсуляція корисного вантажу безпеки ESP (Encapsulation Security Payload) і заголовок автентифікації. У випадку з ESP, можна або інкапсулювати тільки корисне навантаження IP пакету (транспортний режим) або весь IP пакет (в тунельному режимі). Асоціації безпеки SA (Security Association) в IPSec визначає які криптографічні ключі і алгоритми потрібно використовувати. В SA визначається трьома складовими: IP-адреси призначення; ідентифікатор протоколу; індекс параметра безпеки.

Цей односпрямований SA може бути явним чином налаштований, або він може бути створений динамічно, наприклад, шляхом обміну ключами в Інтернеті по протоколу IKEv2. В загальному програма IPSec є віртуальна приватних мереж VPN (Virtual Private Network) для захисту корпоративної мережі. Трафік може бути захищений на транспортному рівні з допомогою захисту транспортного

рівня по протоколу захисту на транспортному рівні TLS (Transport Layer Security), який базується на аналогічному протоколу рівня захищених сокетів SSL (Secure Sockets Layer). Його основне застосування – захист протоколу http поверх TLS/SSL (https), але він також може бути використаний як самостійний протокол.

У протоколи TLS/SSL, включають перевірку автентичності та ключ, утворений на основі цифрових сертифікатів. Останнім часом підтримка надшарових ключів (PSK-TLS) були розроблені та представлені [16]. Це також можливо для захисту трафіку в більш високих шарах. Це дозволяє виконувати операції по забезпеченню безпеки окремого додатка. Наприклад, електронні повідомлення можуть бути зашифровані (захист конфіденційності) та/або підпис (аутентифікація, цілісність і неможливість підроблення даних) за допомогою S/MIME або протоколу достатньо надійної конфіденційності PGP (Pretty Good Privacy).

3.4.4 Mesh-сценарії користувача

Завдання безпеки і відповідні заходи багато в чому залежить від навколишнього середовища/сценаріїв, в якому використовується WMN. Аналіз питань безпеки, відповідних загрозам, та розробки відповідної архітектури безпеки можливе тільки щодо конкретного області застосування Mesh-мереж.

Це можна легко побачити при розгляді різних сценаріїв використання для Mesh-мережі технології:

військова мережа, де навіть мінімальні переміщення повинні бути конфіденційними;

особиста Mesh-мережа, де всі вузли належать одній особі;

оператор-контрольовані мережі, де всі вузли, складові мережу, знаходяться під єдиним адміністративним контролем;

мережа, яка складається з невеликого набору взаємно довірливих користувачі;

мережа передачі даних з де користувачі, не можуть довіряти на всі 100% або не знають один одного.

Це також може бути необхідно, щоб відрізнити різні типи вузлів, наприклад, вузлів під контролем користувача, вузлів інфраструктури для поліпшення покриття мережі або вузлів, які виступають як шлюзи до інших мереж. Може виявитися, що тільки деякі вузли допускається до участі в збереженні інформації маршрутизації або пересилання пакетів/кадрів. Можливо, вузли розрізняються за своїми можливостями щодо користувальницького інтерфейсу, обчислювальні потужності, пам’ять і ємності батареї.

Рис. 3.9 показує відповідні підрозділи WMN, які складаються з Mesh-вузлів. Це можуть бути виділенні mesh-вузли, тобто вузли, які тільки виконують віддалено задачі маршрутизації і перенаправлення трафіку користувача, аутентифікацію тощо. Можуть існувати і чисто вузли користувачів UN (User Node), які використовують послуги зв’язку mesh-мережі, але не є частиною мережі, тобто вони не беруть участь у маршрутизації. Підключення до мережевої інфраструктури забезпечується шлюзовим вузлом GW (Gataway Node). Хоча GW показано, як окрему сутність на рис. 3.8, вона може бути інтегрована у Mesh-вузли MN (Mesh Node) і Mesh-вузли користувачів MUN (Mesh User Node).

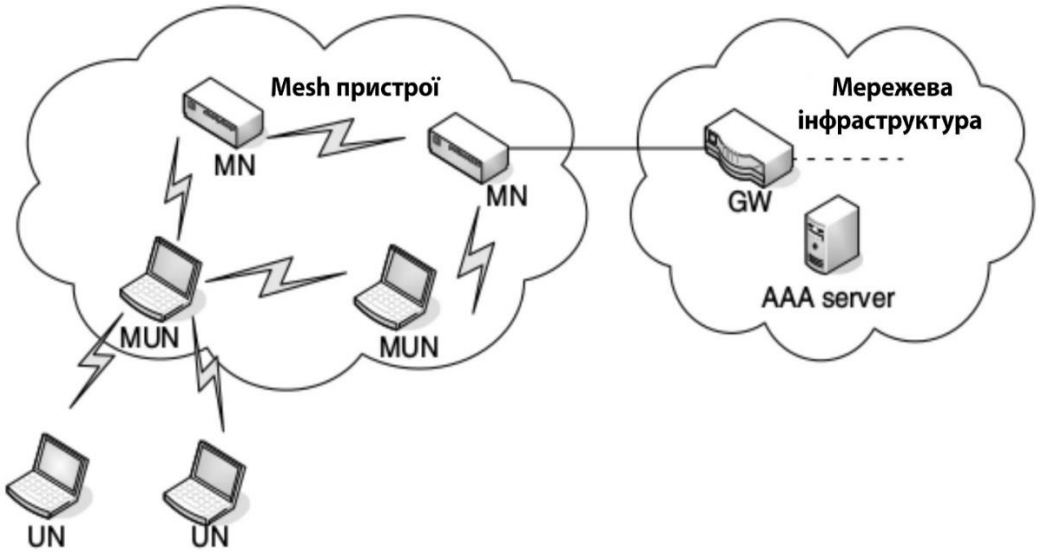


Рис. 3.9 Сценарій Mesh-мережі

У загальному випадку, більш ніж один GW може бути розгорнутий. Показано також сервер AAA, який містить записи для різних сутностей (тобто для користувачів і при необхідності також для MN). В AAA-сервер використовується під час доступу до мережі для аутентифікації користувачів/вузлів так, що приймаються тільки авторизовані. Сервер AAA призначає політику авторизації для різних суб'єктів, що визначають їх роль (дозволи), і він також може бути використаний для обліку.

Фактори розрізнення сценаріїв використання такі:

1. *Адміністративні домени.* У межах одного адміністративного домена, можна припустити, що MN будуть поводити себе коректно. З MN, що належать до декількох адміністративних доменів, деякі MN можуть поводитися некоректно.

2. *Класи користувачів.* Різні класи користувачів можуть бути оброблені по-різному mesh-мережею. Наприклад, коли мережа використовується для публічного доступу і громадської безпеки, трафік всередині мережі повинен бути розділений залежно від користувачів.

3. *Автономні або інтегровані.* Mesh-мережа може бути самостійною, або розгорнутися на існуючій мережевій інфраструктурі. Коли вона інтегрована з мережевою інфраструктурою, то вона може забезпечити безпечну та надійну мережу поверх існуючої.

Багаторівневий захист в Mesh-мережах наведений на рис. 3.10.

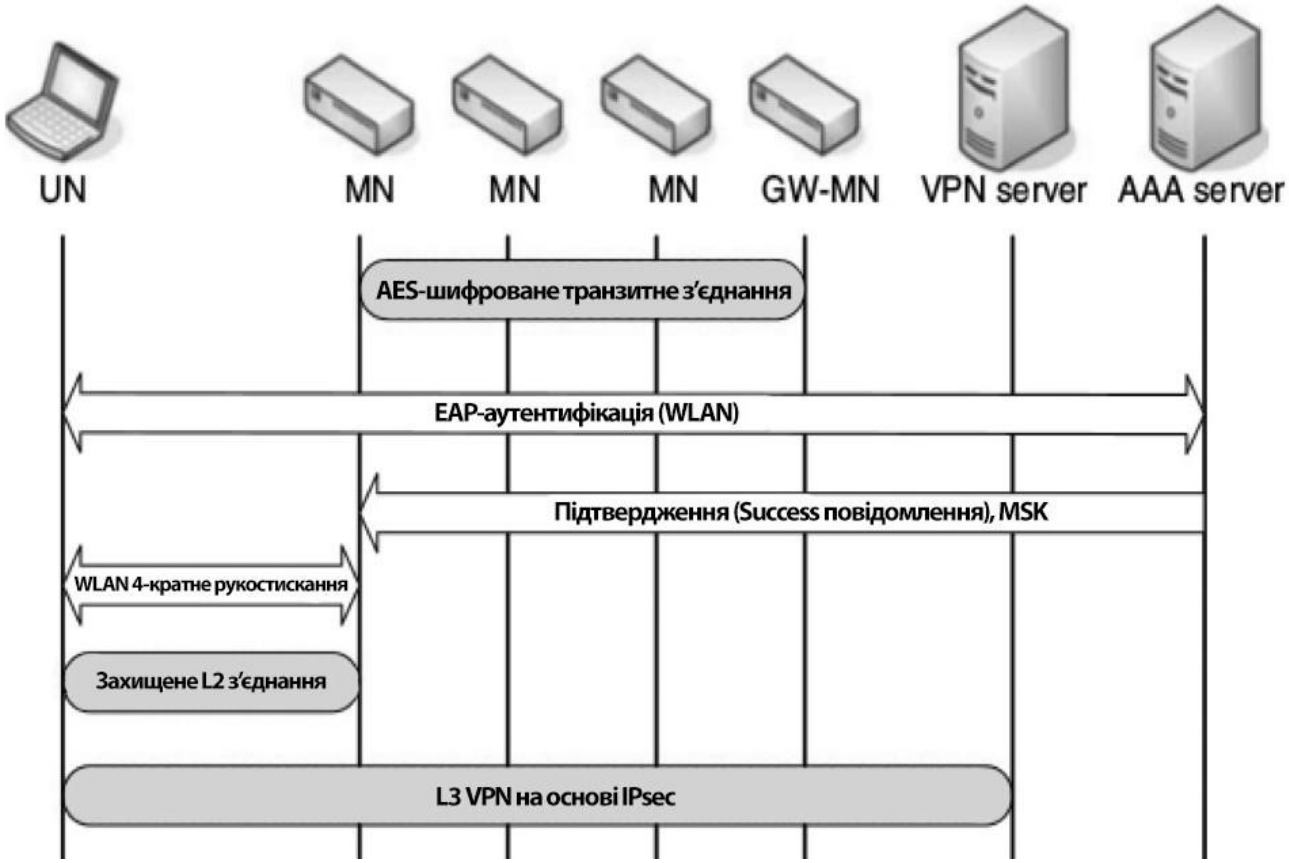


Рис. 3.10 Багаторівневий захист в Mesh-мережах

3.5 Основні дистрибутиви Kubernetes для Edge Computing

Kubernetes початково не був розроблений для роботи із бездротовими стандартами передачі даних, в тому числі Mesh-мережами та IoT (Internet of Things) рішеннями, але сьогодні є дистрибутиви (рішення на основі Kubernetes із певними змінами) саме для тих задач, такі як KubeEdge, k3s, MicroK8s [16–18]. Структурна схема інформаційної системи (рис. 3.11) використовує k3s, який із трьох рішень має найбільшу популярність та виграє за низкою інших характеристик [19].

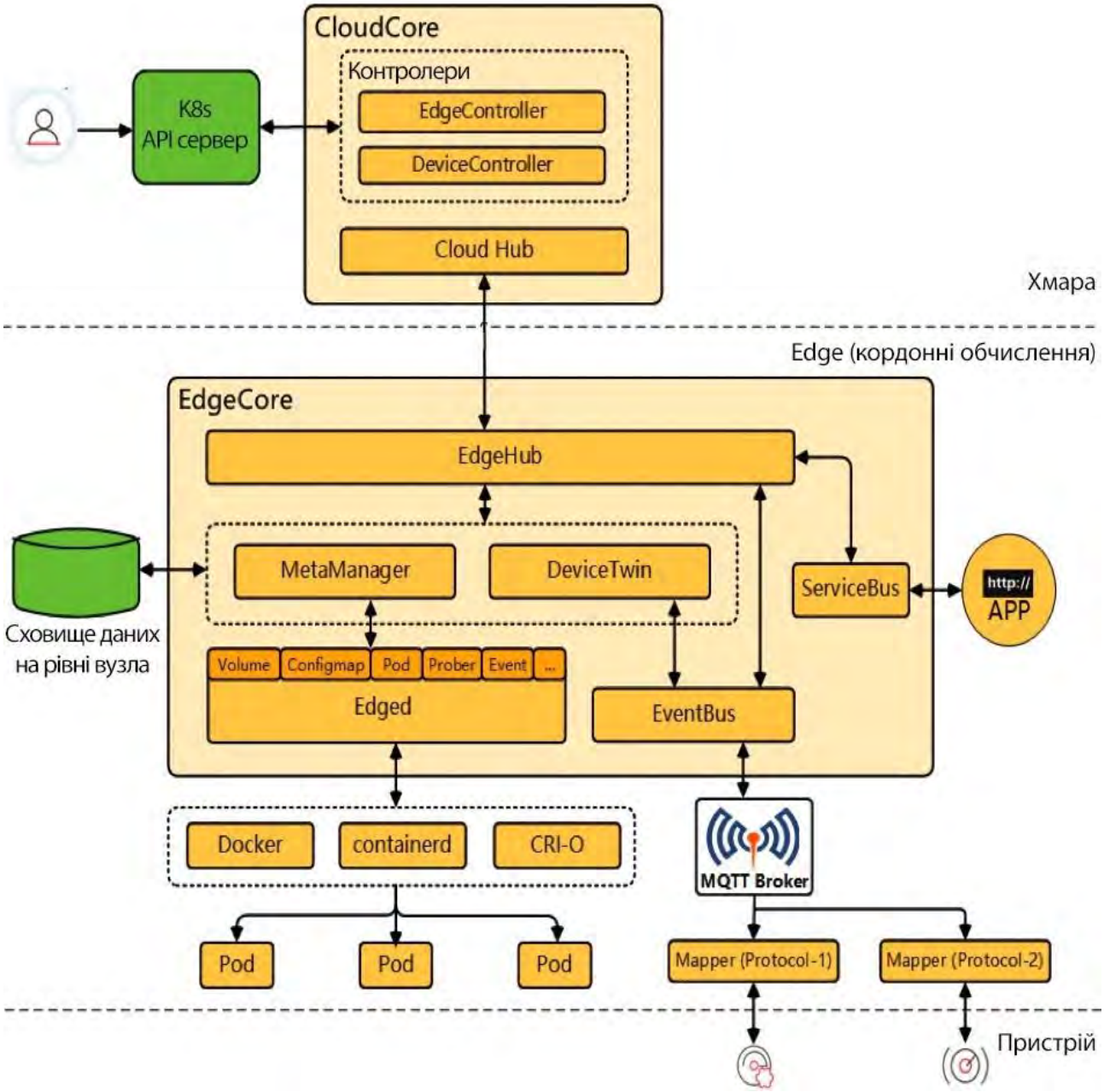


Рис. 3.11 Архітектура KubeEdge

KubeEdge – це система з відкритим кодом, яка розширює оркестровку контейнерних програм і керування пристроями на Edge (кордонні обчислення), вузли. Він створений на основі Kubernetes і забезпечує підтримку основної інфраструктури для мереж, розгортання програм і синхронізації метаданих між Cloud частиною (хмарні обчислення) і Edge. KubeEdge нативно підтримує черги повідомлень телеметричного транспорту MQTT (Message Queue Telemetry

Transport) і дозволяє розробникам створювати спеціальну логіку та забезпечувати зв'язок пристрою з обмеженими ресурсами на Edge.

До переваг цієї системи та Edge computing в цілому можна віднести наступне:

1. *Edge Computing*. Завдяки бізнес-логіці, що працює на edge пристроях, набагато більші обсяги даних можна захищати та обробляти локально, де вони створюються. Це зменшує вимоги до пропускної здатності мережі між Edge і Cloud. Це підвищує швидкість реагування, зменшує витрати та захищає конфіденційність даних.

2. *Спрощення процесу розробки.* Розробники можуть писати стандартні додатки на основі HTTP або MQTT, контейнернезувати їх і запускати будь-де – або на Edge-пристроях, або в Cloud середовище – залежно від того, що більше підходить.

3. *Підтримка Kubernetes*. За допомогою KubeEdge користувачі можуть оркеструвати додатки, керувати пристроями та відстежувати стан додатків і пристроїв на Edge вузлах так само як традиційний кластер Kubernetes у Cloud середовищі.

4. *Підтримка різноманітних додатків*. На Edge при використанні правильного інструментарію легко отримати та розгорнути існуючі складні програми МН, розпізнавання зображень, обробки подій та інші високорівневі програми.

До переваг саме Kubeedge відносно інших схожих інструментів можна віднести підтримку MQTT брокера для управління Edge вузлами. MQTT брокер використовує концепцію передачі інформації на основі черги завдань, перевага якої описана в роботі на дослідження системою керування мобільними агентами [16]. Черга завдань в системі управління мобільними агентами підтримує список завдань, які мають бути виконані мобільними агентами. Цей список упорядкований за пріоритетом або за часом надходження завдання. Також слід відзначити компонент EdgeMesh [20], що є доповненням до KubeEdge, забезпечуючи просте виявлення сервісів (service discovery) і проксі-функції

трафіку для додатків, тим самим захищаючи складну мережеву структуру в Edge сценаріях.

K3s [17] – це високодоступний, сертифікований дистрибутив Kubernetes, розроблений для продуктових навантажень у віддалених місцях без нагляду, з обмеженими ресурсами або для IoT девайсів. Rancher пропонує k3s як полегшений дистрибутив Kubernetes, також з упором на edge сфери застосування. Дистрибутив повністю сумісний з Kubernetes, містить усі базові компоненти за замовчуванням і націлений на швидкий, простий і ефективний спосіб забезпечити високодоступний і відмовостійкий кластер для набору вузлів.

Концепція високої доступності забезпечується використанням декількох машин (3 і більше), що також покращує масштабованість, системи дозволяючи розділити навантаження між вузлами, що показано на рис. 3.11. Високодоступна конфігурація k3s. Для спрощення розгортання та експлуатації використовується один бінарний файл. Систему управління базами даних (СУБД) etcd замінено на sqlite3 для зниження використання ресурсів, але такі СУБД як Etcd3, Mysql, Postgres також підтримуються. Деякі драйвери були видалені для забезпечення малого розміру бінарного файлу, що становить менше 100 МБ.

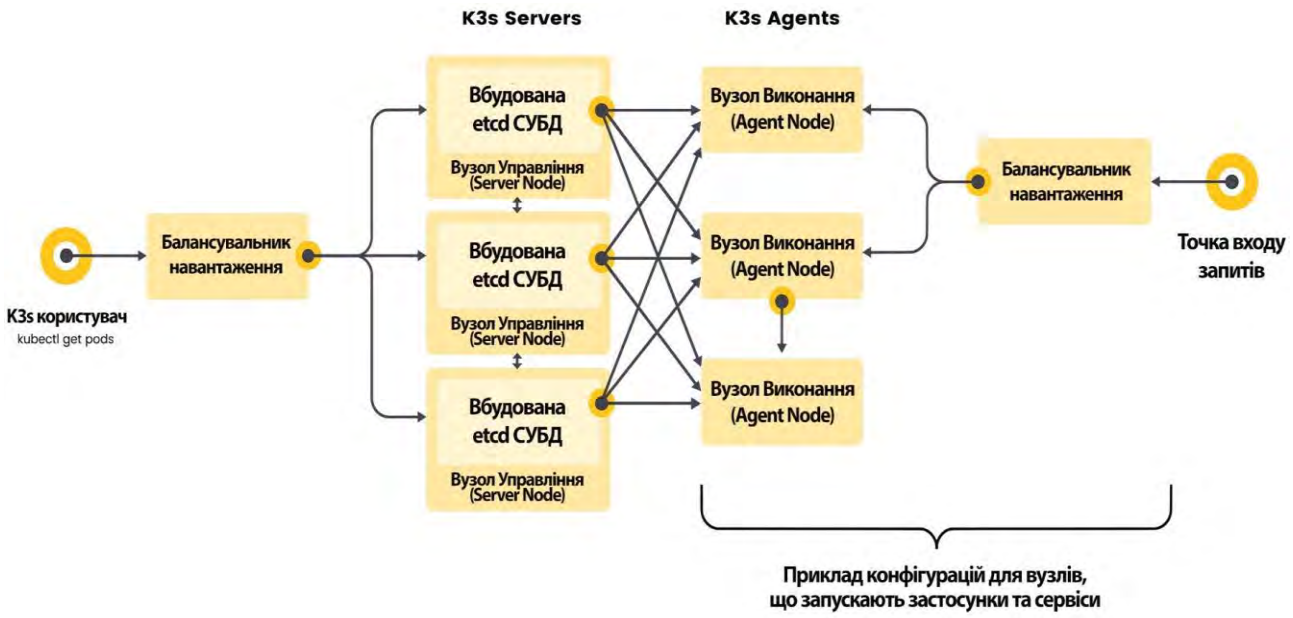


Рис. 3.12 Високодоступна конфігурація k3s

K3s намагається зменшити обсяг пам’яті шляхом реорганізації компонентів рівня керування в кластері. Щоб досягти високої доступності, нові робочі вузли можна легко додати до кластера. Мінімальні вимоги до апаратного забезпечення є 1 CPU і 512 МБ пам’яті.

Ще один із популярних дистрибутивів є microk8s. Його розробка підтримується компанією Canonical. Microk8s спрямований на спрощення використання Kuberntes в загальнодоступних і приватних Cloud середовищ, надаючи легкий і повністю сумісний дистрибутив, особливо для edge галузей застосування, таких як IoT. Однак, все-таки microk8s більше займає нішу дистрибутивів для локальної розробки додатків і тому, показав себе значно гірше в тестах на продуктивність по таким параметрам як швидкість запуску компонентів, використання CPU і оперативної пам’яті, використання дискового простору стосовно k3s, що були проведені в наступному дослідженні [17]. Опираючись на проведені результати, в контексті Edge Computing де пристрої можуть мати обмежені обчислювальні ресурси, k3s є кращим вибором.

Якщо брати до уваги те, що нам потрібно визначитись стосовно оптимального дистрибутиву для мобільних агентів, то між платформи k3s та KubeEdge було надано перевагу k3s, виходячи із наступних чинників:

1. *Спільнота* – k3s і KubeEdge є open-source продуктами та їх розвиток залежить від активності спільноти, які лежить в основі розробки цих інструментів. K3s початково розроблявся компанією Rancher та має біля себе велику англомовну аудиторію. На противагу KubeEdge і особливо компонент EdgeMesh розробляється розробниками з Китаю. Відеоконференції робочих груп здебільшого проходять китайською мовою, а не англійською, що ускладнює більш глибоке вивчення технології.

2. *Автономність*. Kubedge має чітке розділення на Cloud і Edge компоненти, що знижує його автономність та функціональну стійкість. Високодоступна конфігурація k3s.

3. *Гнучкість.* k3s легше більш гнучким в налаштуваннях.

3.6 Особливості реалізацій Mesh-мереж EdgeMesh і Mesh Rider

3.6.1 EdgeMesh мережа

EdgeMesh [20] є нативним доповненням до KubeEdge, однак може працювати і на інших дистрибутивах, забезпечуючи просте виявлення сервісів (service discovery) і проксі-функції трафіку для додатків, тим самим захищаючи складну мережеву структуру в Edge сценаріях. Архітектура високої доступності EdgeMesh в основному спрямована на граничні сценарії, такі як розподілені динамічні ретрансляційні з’єднання і мережева автономія приватних локальних мереж. Він забезпечує ретрансляцію трафіку для вузлів, щоб гарантувати, що кластерні з’єднання будуть працювати в штатному режимі в граничних сценаріях, наприклад коли один чи декілька вузлів вийшли із ладу.

Основна ідея високої доступності EdgeMesh полягає в тому, що коли вузол у кластері має можливість ретрансляції, агент edgemesh візьме на себе роль ретранслятора для забезпечення передачі трафіку до інших вузлів. Коли кластер ініціалізується або новий вузол приєднується до кластера, система EdgeMesh виявляє та записує вузли в локальній базі на основі механізму mDNS (multicast DNS). У той же час механізм розподіленої геш таблиці DHT (Distributed Hash Table) відповідатиме на запити підключення від інших вузлів за межами локальної мережі. Коли два вузли локальної мережі хочуть бути з’єднані, ретрансляційний вузол може надати їм послуги ретрансляції трафіку та сприяння проникненню в інтрамережу.

Концепцію високої доступності EdgeMesh показано на рис. 3.13 [20]. Вузол A з’єднується із вузлом B через проміжний вузол R1 для утворення з’єднання linkab (рис. 3.13, а). Вузол A роз’єднаний із R1 (рис. 3.13, б). Вузол A створює нову з’єднання із проміжним вузлом R2, а з’єднання linkab залишається без змін (рис. 3.13, в).

У кластері сервіс Linkab забезпечується з’єднанням від A до B через ретренслятор R1. Коли R1 не може надавати послуги ретрансляції, вузли A і B автоматично використовуватимуть функцію високо доступності для надання послуг, перемикаючись на ретрансляцію через R2 і повторно встановлюючи

з’єднання. Під час цього процесу користувачі майже не відчувають зміни в роботі сервісу.

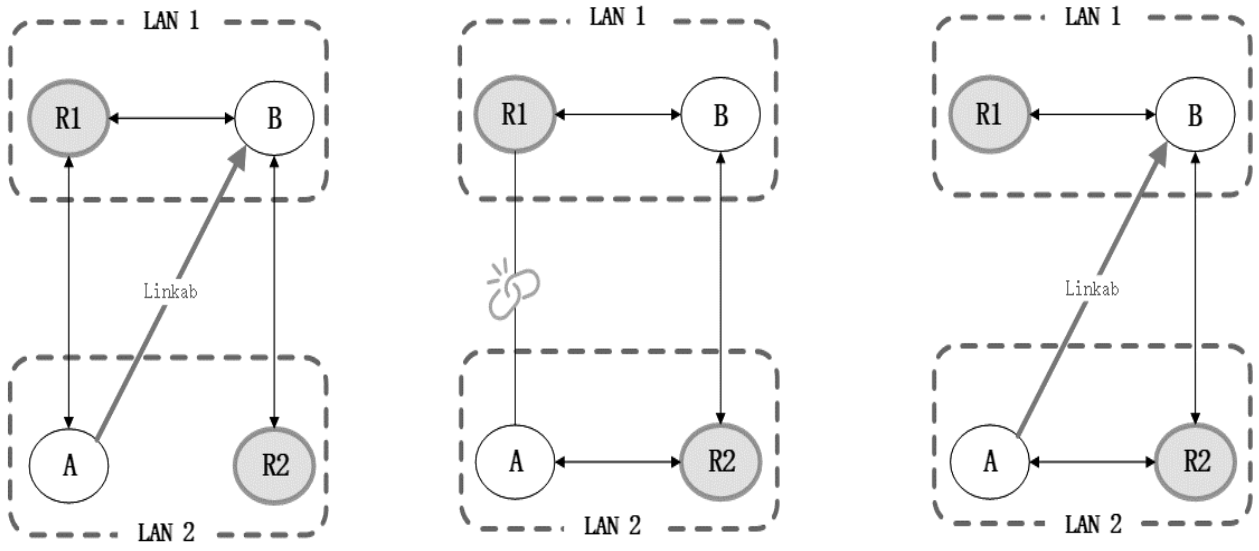


Рис. 3.13 Концепція високої доступності EdgeMesh

Описана система покращує функціональну стабільність мережі за рахунок використання Mesh-архітектури передачі даних. Edgemesh реалізовний на рівні додатків OSI (7 рівень). Завдяки можливості перегляду списку на edge девайсів, EdgeMesh відстежує додавання, видалення та зміну метаданих, а потім створює правила iptables на основі кінцевих вузлів. Правила iptables, попередньо налаштовані EdgeMesh, перенаправлятимуть запити на порт 40001, на якому працює процес EdgeMesh.

Після надходження запитів, EdgeMesh завершує вибір маршрутизації (тут відбувається балансування навантаження), а потім надсилає запити на хост, де розташований кінцевий сервіс. Суттєвим мінусом цієї реалізації є не відповідність стандартам CNI (Container Network Interface), що уніфіковує мережеві розширення в Kubernetes, але на момент написання статті ведеться робота над цим.

3.6.2 Mesh Rider мережа

Mesh Rider – це потужна запатентована мобільна Mesh бездротова широкосмугова технологія, розроблена для випадків мобільного використання. Інноваційна форма сигналу забезпечує надійний канал передачі даних із низькою затримкою та високою пропускною здатністю на надзвичайно великих відстанях. Вбудований інтелект дозволяє йому динамічно адаптуватися до складних радіочастотних сценаріїв, зберігаючи підключення.

Радіоприймач Helix Mesh Rider Radio (HMRR, рис. 3.14) був розроблений частково за підтримки Відділу оборонних інновацій Міністерства оборони США для програми Blue UAS. Радіостанція використовує найсучаснішу технологію, включаючи підтверджене шифрування рівня FIPS 140-3 і запатентовані можливості перемикання діапазонів, що забезпечує безпечний і надійний зв’язок у разі спроб глушіння. Розширені можливості сітчастої мережі забезпечують безперебійне підключення навіть у складних і непередбачуваних середовищах. Розгорнуті в Україні БпЛА з бортовим радіо HMRR покращують тактичну обізнаність команд, збільшують радіус дії та неперевершену стійкість до ворожого втручання.

Рис. 3.14 Радіоприймач Helix Mesh Rider Radio

Винятковий радіус дії та пропускна здатність радіостанції забезпечують безперебійний міжміський зв’язок, забезпечуючи безперебійне з’єднання у

великих зонах роботи. Відданість Doodle Labs своєчасній доставці гарантує українським виробникам БпЛА доступ до критичного обладнання без затримок. Це сучасне рішення для бездротового зв'язку розроблено для того, щоб розширити можливості роботи UAS, підвищити їхні можливості, одночасно забезпечуючи надійну стійкість проти загроз противника. Doodle Labs залишається відданим інноваціям і досконалості в технологіях бездротового зв'язку. Співпрацюючи з українськими виробниками БпЛА, компанія підтверджує свою місію з надання розширених можливостей БпЛА та захисту критичних операцій.

Усвідомлюючи потребу в надійних і стійких системах зв'язку під час операцій БпЛА, Ukrspecsystems інтегрувала перевірену в боях Doodle Labs HMRR в БпЛА Mini Shark. Це потужне рішення бортового каналу передачі даних розроблено спеціально для подолання викликів, з якими стикаються Збройні Сили України. Оператори в Україні повідомляють про можливість успішної передачі HD-відео на відстань до 80 км і уникнення активних спроб російського глушіння, використовуючи шість доступних частотних діапазонів багато діапазонного HMRR. Наявність такого рішення, що вже використовується на полі бою підтверджую можливість та доцільність використання Mesh-мереж.

3.7. Висновки до розділу

В цьому розділі розглянуто безпроводові Mesh-мережі, такі параметри як архітектура мережі, пропускна здатність та обладнання та безпека. Також розглянуто основні проблеми розробки та вимоги, які накладаються на протоколи на різних рівнях OSI, для досягнення максимальної ефективності функціонування WMN. Особливу увагу було приділено стандарту IEEE 802.11s на базі якого і буде змодельовано Mesh-мережу на основі БпЛА.

Актуальність Mesh-мереж визначається розвитком мікроелектроніки, появою безлічі різних пристроїв, здатних працювати автономно довгий час, мають особливість багаторазової зміни режиму (online находження в мережі та offline-виходу з мережі) і потребують обміні інформацією зі своїм оточенням, а

можливо і з керуючим або інформаційним центром (підтримка концепції IoT). Розвиток цих сформував появу нового терміну, як Edge Computing.

Edge Computing – це парадигма розподілених обчислень, яка переносить обчислення і носії даних ближче до джерела даних. Таким чином значно знижується затримка, оскільки дані обробляються на місці їх виникнення. Це особливо важливо для застосунків, які працюють в режимі реального часу, таких як автономні автомобілі та БпЛА, IoT тощо. Парадигма Edge computing також має популярність для платформи Kubernetes, де існує тренд його розгортання повністю або частково на Edge-пристроях.

3.8 Список використаних джерел

1. Zhang Y., Luo J., Hu H., “Wireless mesh networking”, 2016, 610 c.

2. IEEE 802.16, “IEEE Standard for Local and metropolitan area networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems”, IEEE, 2004.

3. IEEE 802.11, “IEEE Standard for Information Technology – Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks – Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE, 2021.

4. X. Yingqing, L. Zhimin, “Design and Implementation of Wireless Mesh Network Testbed Based on Layer 2 Routing”. 4 th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM, 12–14 October 2008, pp. 1–4.

5. G.R. Hiertz, S. Max, T. Junge, D. Denteneert, L. Berlemann, Y. Zang, “IEEE 802.11s MAC Fundamentals” IEEE International Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007, 8-11 October 2007, pp. 1–8.

6. N. Imran, S. Riaz, A. Shaheen, M. Sharif, M. Raza, “Comparative analysis of link state and distance vector routing protocols for mobile AdHoc networks”. Sci. Int. (Lahore), 2014, vol. 26(2), pp. 669–674.

7. G. R. Hiertz et al., “IEEE 802.11s: The WLAN Mesh Standard”. In IEEE Wireless Communications, vol. 17, February 2010, no. 1, pp. 104–111.

8. The complete family of wireless LAN standards: 802.11 a, b, g, j, n. News from Rohde & Schwarz, Number 183 (2004/III), pp. 28–30. [Електронний ресурс] – Режим доступу до ресурсу: https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_news_from_rs/183/n183_lan.pdf (дата звернення 01.11.2023).

9. E. Khorov, A. Kiryanov, A. Lyakhov and G. Bianchi, “A Tutorial on IEEE 802.11ax High Efficiency WLANs”. In IEEE Communications Surveys & Tutorials, vol. 21, 2019, no. 1, pp. 197–216.

10. Барабаш О. В., Кравченко Ю. В., “Функціональна стійкість – властивість складних технічних систем”. Зб. наук. пр. НАОУ, бюл. № 40, К.: НАОУ, 2002, с. 225–229.

11. Hariyanto H., Santoso H., Widiawan A., “Emergency broadband access network using low altitude platform”. Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICIBME), 2009 International Conference on, Nov 2009, pp. 1–6.

12. N. Uchida, M. Kimura, T. Ishida, Y. Shibata, and N. Shiratori, “Evaluation of Wireless Network Communication by Autonomous Flight Wireless Nodes for Resilient Networks”. 2014 17th International Conference “Network-Based Information Systems” (NBiS), Sept 2014, pp. 180–185.

13. Відкритий код для симуляції mesh мережі [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nsnam/ns-3-dev-git/blob/master/src/mesh/examples/mesh.cc (дата звернення 01.11.2023).

14. IEEE 802.11i, “Medium Access Control (MAC) Security Enhancements”. IEEE, 2004.

15. NS-3 network simulator [Електронний ресурс] – Режим доступу до ресур-су: https://www.nsnam.org/ (дата звернення 01.11.2023).

16. Why KubeEdge. [Електронний ресурс] – Режим доступу до ресурсу: https://kubeedge.io/ (дата звернення 01.11.2023).

17. Lightweight Kubernetes. [Електронний ресурс] – Режим доступу до ресурсу: https://k3s.io/ (дата звернення 01.11.2023).

18. The effortless Kubernetes. [Електронний ресурс] – Режим доступу до ресурсу: https://microk8s.io/ (дата звернення 01.11.2023).

19. Кузьміч М. Ю., Гордієнко Т. Б., “Упровадження Kubeflow MLOps у розподілені інформаційні системи мобільних агентів із покращеною функціональною стійкістю”. Зв’язок, 2023, № 6 (166), с. 28–32.

20. EdgeMesh High Availability. [Електронний ресурс] – Режим доступу до ресурсу: https://edgemesh.netlify.app/guide/ha.html#using-ha-feature (дата звернення 01.11.2023).

РОЗДІЛ 4
РЕЗУЛЬТАТИ ВПРОВАДЖЕННЯ ПІДХОДУ MLOps ІЗ ВИКОРИСТАННЯМ Kubeflow ТА Kubernetes

4.1 Впровадження системи машинного навчання для мобільних агентів

Результати проведених досліджень вказують на те, що Kubeflow складається із різноманітних компонентів з відкритим кодом, які високо інтегровані через платформу Kubernetes. Встановлена висока ефективність Kubeflow у використанні патернів операторів Kubernetes для обробки об’єктів МН. Розробка коду моделі є лише невеликою частиною в широкому спектрі завдань МН, що підкреслює актуальність автоматизації. Водночас доведено, що наявність повноцінного конвеєра для неперервної інтеграції та постачання додатків є ключовою для успішної реалізації цього процесу від початкової розробки до кінцевого користувача.

Практичне застосування зазначеного інструменту можна знайти в багатьох галузях через його гнучкість і універсальність. На основі проведених досліджень було сформовано концепцію інформаційного рішення на основі штучного інтелекту для виявлення ворожих цілей із мобільних агентів – безпілотних літальних апаратів (БпЛА). Системи відеоспостереження БпЛА допомагають військовим стежити за землею, надаючи візуальну інформацію в реальному часі для визначення ситуації та прийняття рішень.

Перелік основних переваг таких системи відеоспостереження наступний:

1. *Моніторинг* – камери спостереження забезпечують постійний моніторинг ключових територій, баз і кордонів.

2. *Ситуаційна обізнаність* – трансляції відео в реальному часі забезпечують розуміння поточних дій і потенційних загроз у реальному часі.

3. *Віддалене спостереження* – камери у віддалених або небезпечних зонах забезпечують видимість без безпосередньої присутності людини.

4. *Безпека периметра* – камери відстежують і сповіщають про несанкціоновані переміщення поблизу периметра.

5. *Ідентифікація цілей* – камери високої роздільної здатності допомагають ідентифікувати потенційні цілі або підозрілу діяльність.

6. *Розвідка* – відео з камер спостереження допомагає збирати розвіддані та планувати військові операції.

7. *Збір доказів* – записаний матеріал може слугувати доказом для розслідувань або дій.

8. *Силовий захист* – камери підвищують безпеку персоналу та активів, надаючи розширені попередження.

9. *Зв'язок* – відеоканали забезпечують кращий зв'язок між різними підрозділами та командними центрами.

10. *Повітряне спостереження* – БпЛА, оснащені камерами, пропонують огляд з висоти для покращеного покриття.

11. *Координація реагування* – системи спостереження допомагають координувати дії на ситуації, що з'являється.

12. *Аналіз після місії* – відео допомагає проаналізувати минулі дії для тактичних і стратегічних удосконалень.

Таким чином, системи відеоспостереження БпЛА підтримують зусилля військових, надаючи важливу візуальну інформацію, підвищуючи обізнаність і сприяючи прийняттю рішень у різних оперативних сценаріях.

В останні роки БпЛА стали невід'ємною частиною звичайних бойових дій. Війна Росії проти України також показує, що ці дистанційно керовані мобільні агенти є необхідними для досягнення перемоги в сучасних конфліктах [1]. НАТО класифікує безпілотники на три загальноприйняті основні типи на основі їх максимальної злітної ваги (табл. 4.1) [2]: безпілотники класу I вагою менше 150 кілограмів (кг), включаючи малі, міні- та мікродрони; тактичні дрони класу II вагою від 150 до 600 кг; і дрони класу III вагою понад 600 кг.

Таблиця 4.1 Класифікація БпЛА НАТО

Клас	Категорія	Висота застосування	Радіус дії	Рівень застосування	Приклад
Клас I (< 150 кг)	малі (> 15 кг)	до 1500 м	до 50 км	батальон, полк	Scan Eagle
	мані (< 15 кг)	до 900 м	до 25 км	рота, взвод, відділення	Skylark
	мікро (< 66 Дж)	до 60 м	до 5 км	взвод, відділення, індивідуально	Black Hornet Nano
Клас II (150–600 кг)	тактичні	до 5,5 км	до 200 км	бригада	Hermes 450
Клас III (> 600 кг)	ударні/ бойові	до 20 км	до 6000 км	театр бойових дій	MQ-9 Reaper
	MALE	до 13,7 км	необмежено	спеціальна цільова група	Heron
	HALE	до 19,8 км		театр бойових дій	Global Nawk

Динаміка безпілотників в Україні в основному демонструє системи I і III класу. У той час як великі безпілотні літальні апарати з ракетами можуть бути руйнівними в умовах переваги в повітрі, малі безпілотні літальні апарати виявляються вирішальними для поінформованості про ситуації на полі бою. Крім того, недорогі ударні дрони «камікадзе» пропонують ще один спосіб доставлення вибухівки.

Всі ці БпЛА, які дуже активно використовуються на лінії зіткнення генерують величезний пласт інформації для ефективного аналізу якого потрібно затрати дуже багато ресурсів. Доцільність використання моделей МН є

запорукою ефективного аналізу, особливо якщо ми хочемо генерувати рекомендації на основі цих даних в режимі реального часу або із невеликою затримкою.

4.2 Етапи машинного навчання конвеєру в розрізі моделі розпізнавання об'єктів

На рис. 4.1 наведено основні етапи МН конвеєру із можливістю постійного навчання для БпЛА [3].

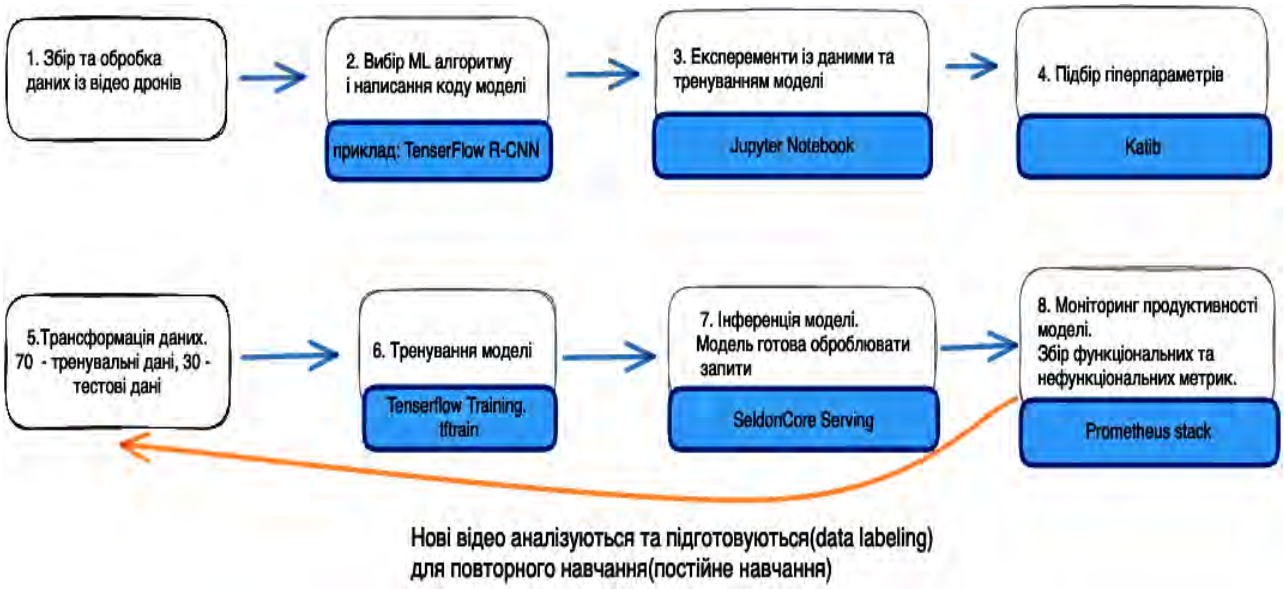


Рис. 4.1 Основні етапи МН конвеєру для БпЛА

Основі етапи МН конвеєру в розрізі пропонованої моделі [3]:

1. *Збір та обробка даних із відео дронів* – відеоматеріали різної якості та із різних джерел, а пріоритетні цілі покадрово визначаються оператором, на основні чого буде навчатись модель.

2. *Вибір ML алгоритму і написання коду моделі* – для пропонованого варіанту підійде модель на основі R-CNN (Region-Based CNN), що дозволяє розв'язувати задачу класифікації об'єктів на фото чи відео.

3. *Експерименти із даними та тренування моделі* – використовуються різноманітні алгоритми та оброблені дані для тренування моделі.

4. *Підбір гіперпараметрів* – задача МН по вибору множини оптимальних гіперпараметрів для алгоритму МН. Гіперпараметр є параметром, значення якого використовується для керування процесом навчання на відміну від значень інших параметрів (наприклад, вагових коефіцієнтів), які потрібно вивчити.

5. *Трансформація даних* – перетворення даних у формат, який потрібний системі навчання. Щоб переконатися, що модель поводиться узгоджено під час навчання та прогнозування, процес трансформації має бути однаковим на експериментальній і продуктовій фазах. Для цього процесу доцільно використовувати такі компоненти як kubeflow pipelies і Jupiter Notebook.

6. *Тренування моделі* – це процес, у якому алгоритм МН подається навчальними даними, з яких він може навчатися. В цьому випадку для тренування доцільно використати компонент Kubeflow tftrain (Tenserflow training). Однієї із переваг tfrain є наявність розподіленого навчання, що дозволяє розподілити навантаження навчання на декілька серверів кластера. З допомогою цього можна значно скоротити час навчання. Потенційне перенавчання моделі можна виконувати на регулярній основі аби адаптовуватись до змін та постійно покращувати ефективність.

7. *Інференція моделі* – це процес запуску моделі в продуктове середовище зі використанням SeldonCore serving. Платформа Kubernetes, на базі якої працює SeldonCore serving, дозволяє гнучко змінювати кількість робочих реплікацій для обробки високого навантаження. Модель готова оброблювати запити.

8. *Моніторинг продуктивності моделі* – це етап, на якому здійснюється хзбір функціональних і нефункціональних метрик. Для цього використовується компонент Prometeus stack.

4.3 Функціональні моделі процесів AS-IS, TO-BE і SHOULD-BE

4.3.1 Загальна характеристика функціональних моделей процесів

Метою побудови функціональних моделей зазвичай є визначення найбільш слабких і уразливих місць в діяльності організації, аналіз переваг нових бізнес-процесів і ступеня необхідних змін наявної структури бізнесу.

Аналіз недоліків і «вузьких місць» починається з побудови моделі AS-IS (як є). Ця модель будується на основі вивчення документації (посадових інструкцій, наказів, звітів тощо), анкетування та опитування службовців підприємства, протоколювання дій співробітників протягом робочого дня та інших джерел. Отримана модель AS-IS служить для виявлення некерованих і не забезпечених ресурсами робіт, непотрібних, неефективних і дублюючих один одного дій та інших недоліків в організації діяльності підприємства.

Виправлення недоліків, перенаправлення інформаційних і матеріальних потоків призводить до створення моделі TO-BE (як буде). Як правило, будується кілька моделей TO-BE, серед яких вибирається найкращий варіант.

Поширена помилка при моделюванні – це створення моделі, що ідеалізується. Прикладом може служити моделювання на основі знань керівника, а не конкретного виконавця робіт. Керівник знайомий з тим, як передбачається виконувати роботи згідно посібникам і посадових інструкцій, і часто не знає, як насправді підлеглі виконують рутинні роботи. В результаті виходить прикрашена, перекручена модель, яка несе неправдиву інформацію і яку неможливо використовувати в подальшому для аналізу. Така модель називається SHOULD-BE (як повинно бути).

Технологія проектування інформаційних систем має на увазі спочатку створення моделі AS-IS, її аналіз і поліпшення бізнес-процесів. Потім створюється модель TO-BE і на її основі будується модель даних, прототип а потім і остаточний варіант інформаційної системи. Побудова системи на основі моделі AS-IS призводить до автоматизації підприємства за принципом «все залишити як є, тільки щоб комп'ютери стояли». Іншими словами автоматизуються недосконалі бізнес-процеси і дублюється існуючий документообіг.

4.3.2 Опис функціональної моделі процесу AS-IS

Відео із БпЛА (рис. 4.2) транслюється його оператором до штабу командування (2 лінія), яке аналізується на місці в ручному режимі. Потім ці дані

у ручному режиму (за необхідності) передаються на рівень вище і вже після того, можуть бути передані сусіднім підрозділам.

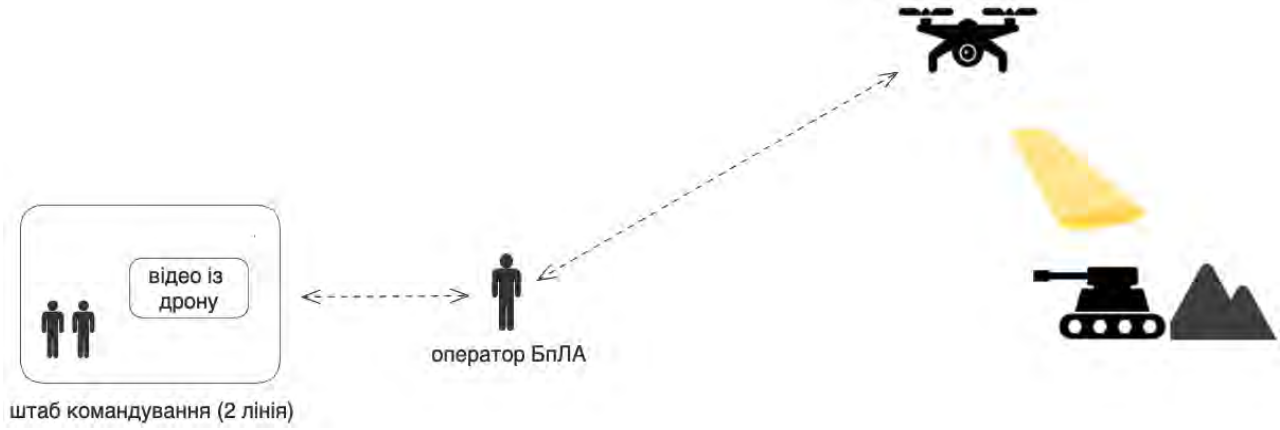


Рис. 4.2 Схематичне зображення функціональної моделі рішення AS-IS

До недоліків такої системи, де відсутнє централізоване сховище даних та автоматизована система їх обробки, можна віднести наступне:

1. *Розділеність даних*. Відсутність централізованого сховища може вести до того, що дані розподілені по різних джерелах і важко доступні для інших компонентів системи. Це може призвести до поганого обміну інформацією між різними частинами системи.

2. *Підвищений ризик помилок*. Якщо дані вводяться вручну, без автоматизованої системи, існує великий ризик помилок та неточностей у введених даних. Це може призвести до невірної інтерпретації та прийняття неправильних рішень.

3. *Низька ефективність*. Автоматизована система обробки даних може значно зменшити час, необхідний для обробки та аналізу інформації. Відсутність такої системи може вести до зайвого витрат часу на ручну обробку даних.

4. *Брак системної цілісності*. Важкість забезпечення цілісності – відсутність централізованого сховища може ускладнити забезпечення цілісності даних. Дані можуть бути розсіяні, і контроль їх цілісності може бути ускладнений.

5. *Обмежені можливості аналітики*. Відсутність централізованої системи може обмежити можливості аналізу та витягування цінної інформації з даних. Аналітика може бути ускладненою через розподіленість і різнорідність джерел даних, тому виникає низька аналітична здатність.

6. *Низька безпека даних.* Розподіленість даних на різних неконтрольованих пристроях може збільшити ризик безпеки даних чи вразливості до атак, оскільки даним важче контролювати та встановлювати засоби захисту.

7. *Велика трудомісткість розвитку*. Додавання нових функціональностей та розширення системи може бути складним через відсутність централізованої архітектури. Це може призвести до великої трудомісткості розвитку системи.

4.3.3 Опис функціональної моделі процесу TO-BE

Інформація (відео) із БпЛА в автоматичному режимі передається до віддаленого безпечного сховища даних від штабу командування або від оператора БпЛА (рис. 4.3). Використовуються різні методи аналізу та роботи із даними як показано на рис. 4.3. Структурна схема системи аналізу даних складається із джерела даних і сховища даних, артефакту аналітики і артефакту МН.

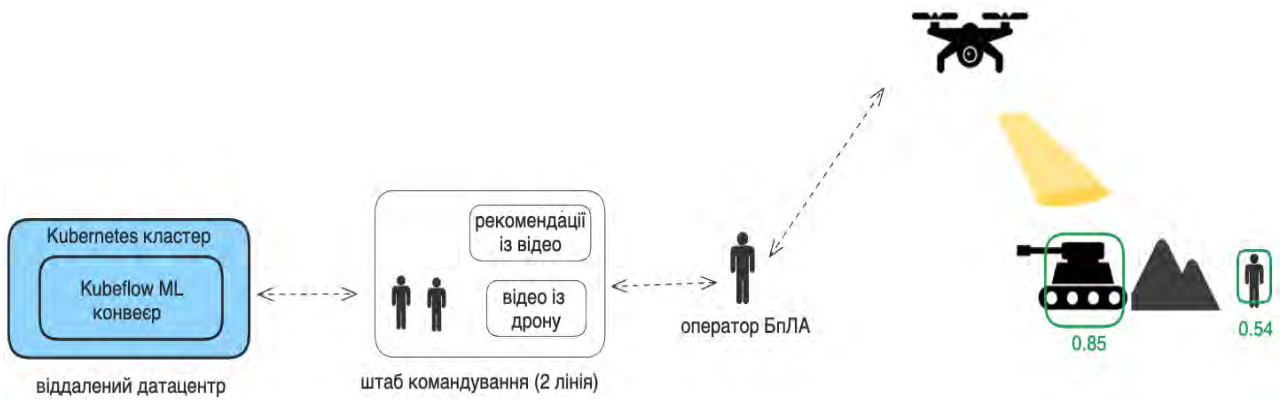


Рис. 4.3 Схематичне зображення функціональної моделі рішення TO-BE

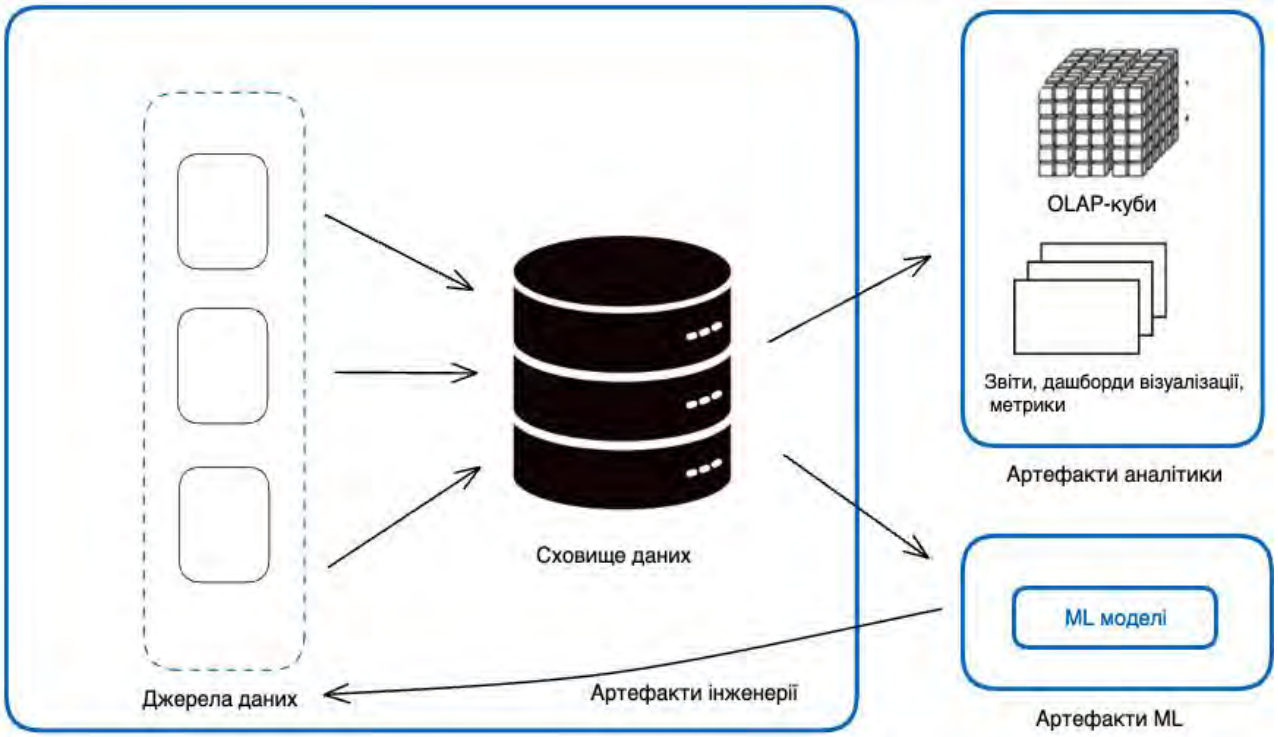


Рис. 4.4 Структурна схема системи аналізу даних

Модель МН розпізнавання об’єктів на основі алгоритму глибинного навчання R-CNN, буде знаходити та розпізнавати об’єкти на відео і надавати рекомендації до штабу командування та оператору БпЛА (рис. 4.5) [3].

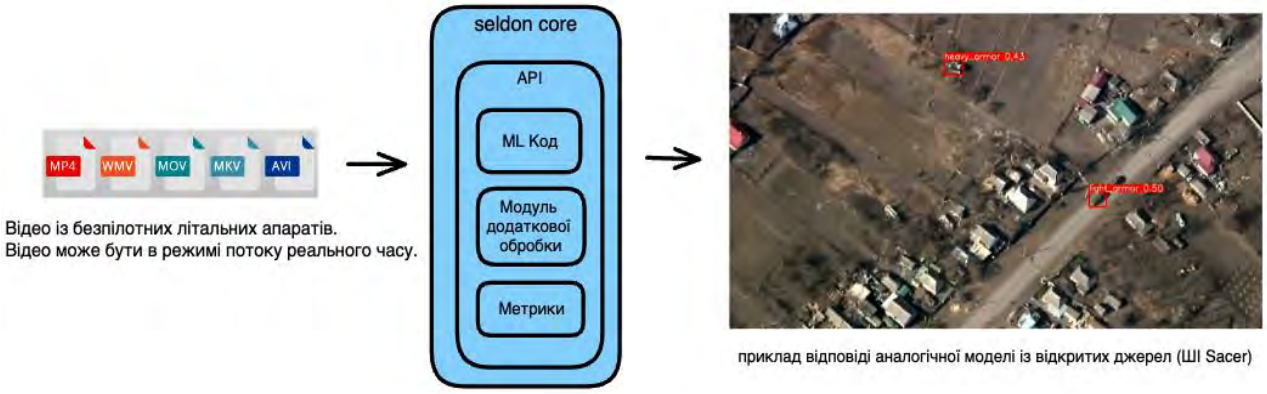


Рис. 4.5 Схематичне зображення інференції моделі із Selden Core

R-CNN є алгоритмом глибокого навчання. R-CNN та його варіації, такі як Fast R-CNN та Faster R-CNN, використовують зверточні нейронні мережі (CNNs)

для задачі об’єктного виявлення на зображеннях. Основна ідея R-CNN полягає у використанні CNNs для виявлення областей (region proposals), які можуть містити об’єкти, а потім використовується вектор класифікації та регресії для точного визначення класу та положення об’єкта всередині кожної області. Такий підхід робить R-CNN ефективним для завдань виявлення та класифікації об'єктів на зображеннях.

При розробці конвеєра MLOps не обов’язково використовувати прив’язку тільки до одного алгоритму та архітектури. Наприклад для підвищення швидкодії можна використовувати архітектуру YOLO (YOLOv4 алгоритм). Кожне сімейство алгоритмів має свої плюси та мінуси. За рахунок підвищеної швидкодії YOLO підходить для задач виявлення об’єктів в реальному часі, тоді як R-CNN може бути вибором для додатків, де важлива висока точність виявлення.

Якщо можливо перевикористати вже наявну модель, при наявності артефактів для її навчання, як висхідний код та дані, на яких вона була натренована, то це доцільно зробити. Доповненням до поточної функціональності може бути модель, що буде визначати точні координати об’єктів на відео, що буде корисним для подальшої візуалізації на карті у випадках коли GPS (Global Position System) сигнал повністю або частково придушений.

МН конвеєр Kubeflow (рис. 4.5), що покриває основні етапи життєдіяльності системи та побудований із використання кращих підходів MLOps дозволить досягти наступних переваг:

1. *Швидше впровадження моделей*. MLOps допомагає зменшити час від розробки моделі до її впровадження в продуктивне середовище. Автоматизація та стандартизація процесів дозволяють ефективно впроваджувати моделі.

2. *Ефективність розробки*. MLOps може спростити та полегшити процес розробки моделей шляхом автоматизації завдань.

3. *Збільшення якості моделей*. Використання MLOps може покращити якість моделей через ефективний моніторинг та як наслідок краще

відлагодження. Збирання та аналіз даних про ефективність моделей у реальному часі дозволяє вчасно виявляти проблеми та вдосконалювати моделі.

4. *Керована безпека та відповідність*. MLOps дозволяє керувати безпекою та дотриманням відповідності, що важливо у випадку використання моделей в регульованому середовищі або при обробці конфіденційної інформації.

5. *Співпраця команд*. MLOps сприяє співпраці між різними командами, такими як розробка, операції та аналітика. Це допомагає створити єдину екосистему для розробки та експлуатації моделей.

6. *Економія ресурсів.* Автоматизація деяких завдань та ефективне використання ресурсів можуть призвести до економії часу та грошей.

7. *Масштабованість*. MLOps надає засоби для масштабування розробки та впровадження моделей великому обсязі.

В контексті поточної задачі для впровадження та розробки моделей для мобільних агентів (БПлА), слід наголосити на таких пунктах, як «Швидше впровадження моделей» і «Ефективність розробки». Тобто, при досягненні певного рівня зрілості конвеєра розробки можна досягти ефекту «постійного навчання», коли модель МН може швидко та якісно оновлюватись відповідно до змін навколишнього середовища. Це є важливим в контексті сучасних бойових дій, оскільки той, хто буде більш гнучким і зможе швидше реагувати на зміни на полі бою матиме перевагу.

4.3.4 Опис функціональної моделі процесу SHOULD-BE

Модель рішення SHOULD-BE (рис. 4.6) має весь функціонал та переваги системи TO-BE, але із певними доповненнями. БпЛА представлені у вигляді мобільних агентів, що використовуються для інференсу моделі. Мобільні агенти формують розподілену інформаційну систему на базу дистрибува Kubernetes k3s, що чудово підходить для Edge Computing.

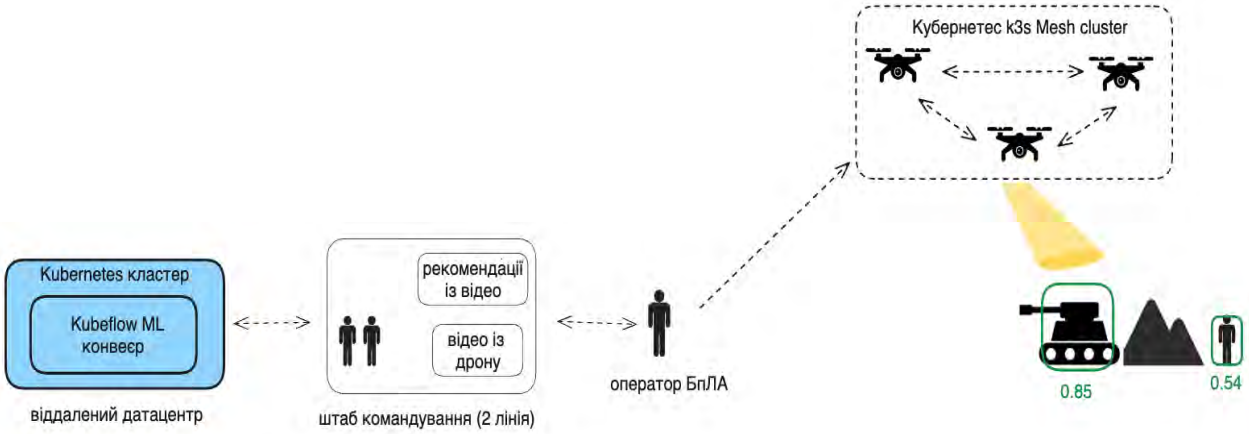


Рис. 4.6 Схематичне зображення функціональної системи SHOULD-BE

Окрім k3s, були розглянуті та інші дистрибутиви (microk8s, KubeEdge). Доцільність та переваги k3s такі: гнучкість, автономність, використання ресурсів, спільнота розробників та користувачів.

Етапи МН конвеєру із можливістю постійного навчання для БпЛА із використанням Edge Computing зображені на рис. 4.7.

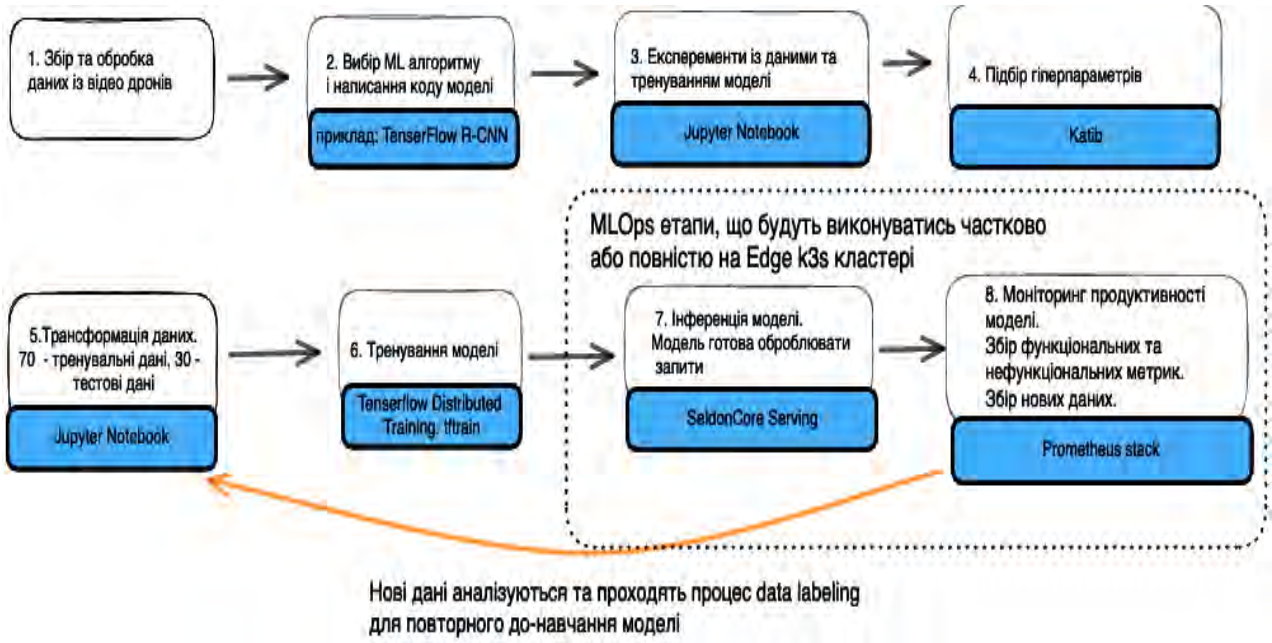


Рис. 4.7 Основні етапи МН конвеєру для БпЛА із використанням Edge computing

4.4 Використання Mesh-мереж для мобільних агентів

Мобільність та динамічність БпЛА робить стандартні мережі неефективними коли потрібно управляти декількома вузлами одночасно. Доцільно розглядати мобільні агенти (БпЛА) частиною MLOps конвеєру і обмін даних між вузлами є необхідною умовою розподілених обчислень. Для забезпечення спілкування між вузлами mesh мережі підходять як найкраще.

Використання mesh-мереж, де кожен вузол може бути підключений до будь-якого іншого, для мобільних агентів, як БпЛА, має наступні переваги:

1. *Гнучкість і масштабованість*. Mesh-мережі дозволяють БпЛА створювати автономні мережі без необхідності підключення до центральної точки доступу. Це дозволяє легко розширювати мережу для охоплення великої площі або додавати нові БпЛА без значного впливу на систему.

2. *Стійкість і надійність*. У випадку використання mesh-мережі, втрата одного з’єднання не призводить до втрати зв’язку. Цей принцип стійкості забезпечує надійну комунікацію між БпЛА, навіть якщо деякі елементи мережі виходять з ладу або втрачають зв’язок.

3. *Можливість обходу перешкод*. Mesh-мережі можуть автоматично знаходити оптимальний шлях для передачі даних, обходячи можливі перешкоди або зони зі слабким сигналом. Це особливо корисно для БпЛА, які можуть опинитися в різних умовах довкілля. На рис. 4.8 наведено можливість інформативного сигналу обходити перешкоди сигналів радіоелектронної боротьби.

4. *Зменшення затримок.* Mesh-мережі можуть допомогти зменшити затримки в передачі даних, оскільки вони дозволяють даним обходити «вузькі місця» і знаходити найшвидший шлях до призначення.

БпЛА управління (Server) – це БпЛА, що володіють достатньої кількістю ресурсів для виконання службових задач розподіленої системи Kubernetes та Kubeflow компонентів для наприклад інференції моделі, як Seldon Core та/або запуску інших ресурсоємних застосунків для контролю над всієї системою. Відповідно вартість цих БпЛА буде вища. Доцільно мати хоча б три БпЛА

управління, для досягнення високої доступності та можливості розподілення навантаження між вузлами. БпЛА специфічних задач (Agent) та БпЛА ретранслятори (Agent) – це службові БПлА, що володіють слабшими характеристиками, ніж службові та виконують вузькоспеціалізовані завдання, що будуть надіслані до них службовим вузлом.

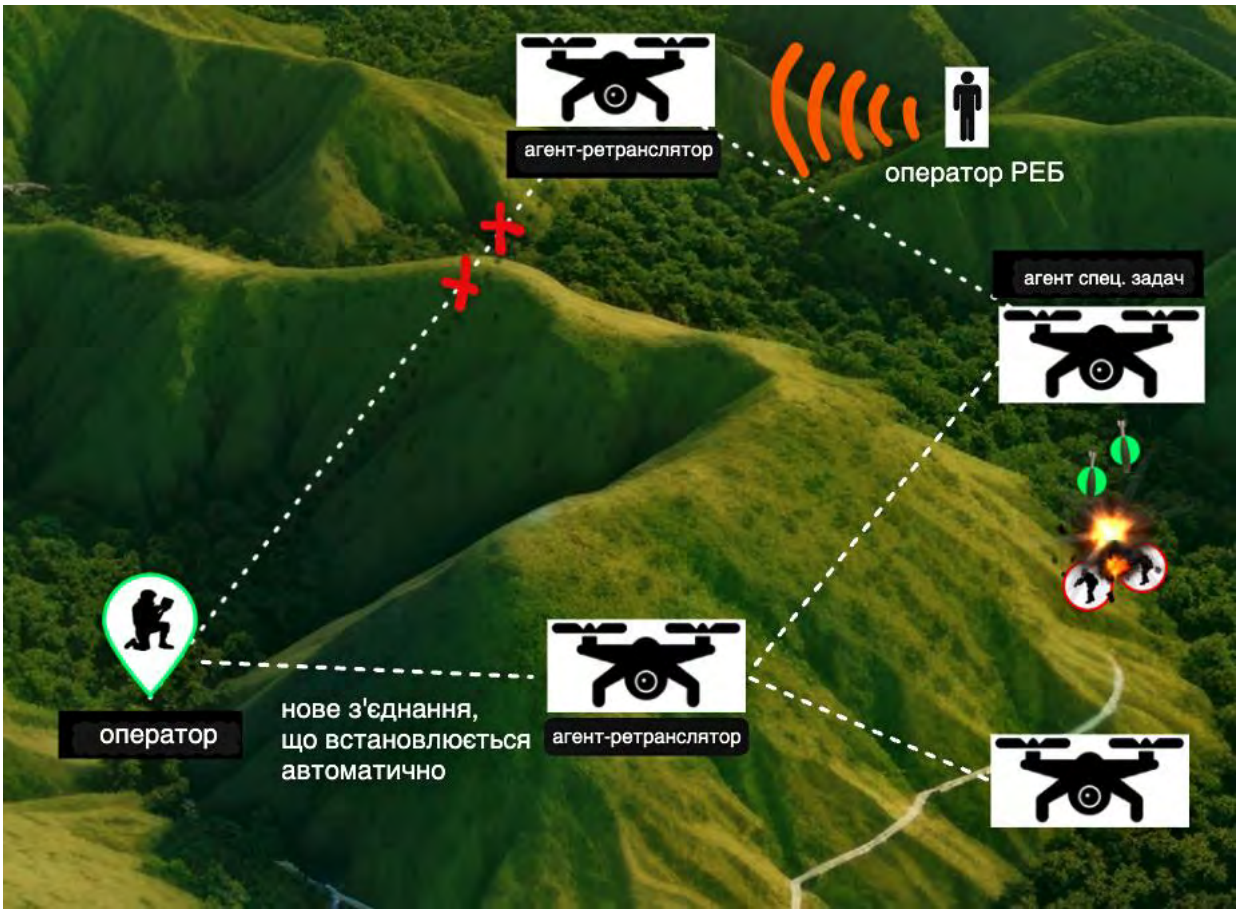


Рис. 4.8 Приклад використання Mesh-мережі для БпЛА
із можливістю протидіяти перешкодам

Прикладом використання цієї концепції може бути наступне: БпЛА специфічних задач виконує функції баражуючого боєприпаса, де модель вибору та розпізнавання і доведення до цілі виконуються на БПлА управління, а надійність та стійкість мережі забезпечується використанням БпЛА ретрансляторів. Тобто наступна система дозволяє знизити собівартість

«розумного» баражуючого боєприпасу, перемістивши роботу «штучного інтелекту» на інший вузол, що може бути використаний багаторазово.

Структурна схема інформаційної системи буде використовувати k3s, який із трьох рішень є найбільш доцільним для використання в розробленій системі [7–11]. Як і Kubernetes, k3s має схожу архітектуру розділення на службові ноди (Server) та ноди працівники (Agent). На Server ноди встановлені необхідні службові компоненти для роботи кластеру та Fgent-компонети, що дозволяють їм запускати клієнтські застосунки також. На Agent-ноди службові компоненти не встановлюються, тільки базові як показано на рис. 4.9 [4].

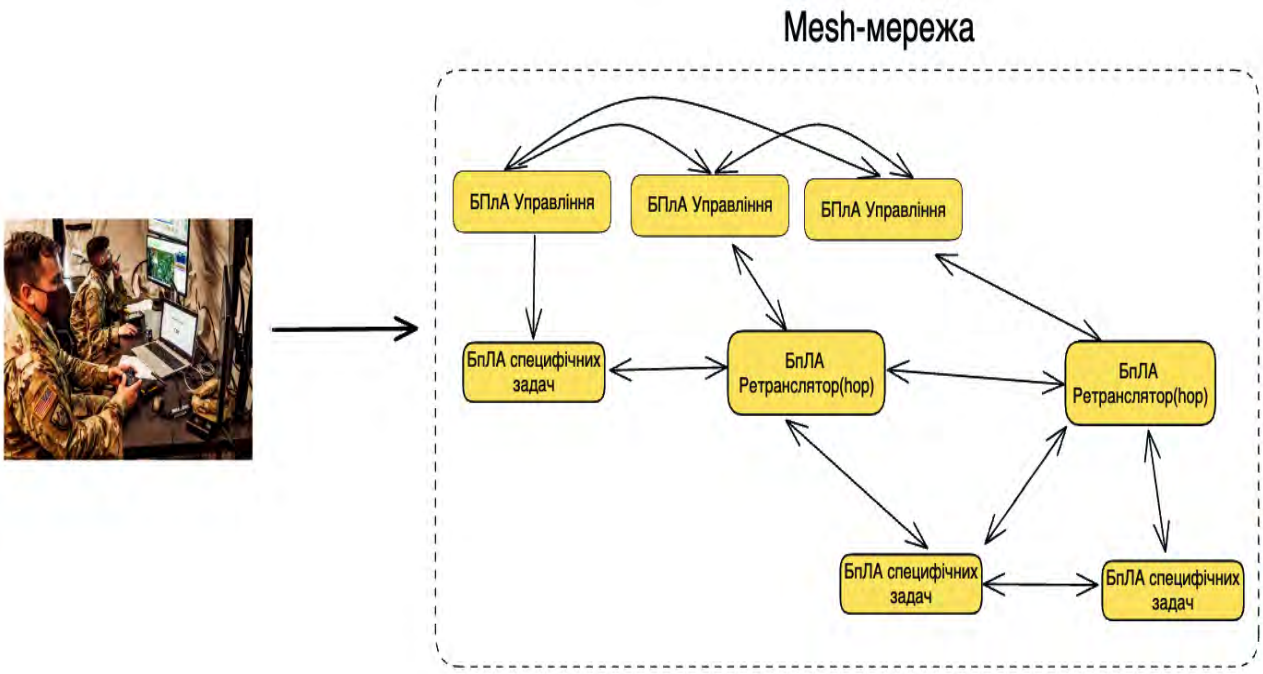


Рис. 4.9 Схематичне зображення розподіленої інформаційної Mesh системи

По реалізації Mesh-мереж для мобільних агентів з урахуванням потенційного подвійного призначення застосовуються:

1. *Стандарт 802.11s*. Підходить для цивільного використання, а для використання у військових цілях потребує доопрацювання. Цей стандарт належить до канального рівня OSI (Open Systems Interconnection), що надає певну перевагу в продуктивності, але зменшує гнучкість.

2. *EdgeMesh*. Реалізація Mesh-мережі на прикладному рівні OSI. До плюсів можна віднести наявність відкритого коду та висока інтеграція із Kubernetes. До мінусів можна віднести обмежену кількість матеріалів англійською мовою (в більшості – китайська) та пасивну спільноту розробників.

3. *Готові рішення*. Mesh Rider Radios від doodlelabs [5]. Doodle Labs – це компанія, що спеціалізується на розробці та виробництві бездротових мережевих рішень, зокрема, мережевих рішень на основі технології Mesh. Їхні вироби часто використовуються в застосуваннях, де важливо створити стійку та ефективну мережу, особливо в умовах, коли інфраструктура та зв'язок можуть бути обмеженими чи важко досяжними.

До плюсів можна віднести те, що Mesh технологія використовується в передових розробках [6], що фінансуються військовим комплексом США, і у БпЛА українського виробництва компанії Ukrspecsystems (Ukrspecsystems). Ця реалізація є закритою, але виходячи із специфікацій можна зробити припущення, що Mesh-мережу реалізована на прикладному рівні (рівень додатків) OSI, по аналогії до EdgeMesh.

4.5 Функціональна стійкість Mesh-мережі на основі мобільних агентів

4.5.1 Особливості існуючих рішень

Сьогодні Mesh-мережі активно використовуються для надання доступу до глобальної мережі там, де немає можливості вирішити цю проблему з допомогою стандартних методів. Розглянемо можливість створення тимчасової мережі передачі даних на основі низько розташованих платформ LAP (Low Atitude Platform). Ця система складається із БпЛА, оснащених дводіапазонною точкою доступу Wi-Fi з функцією Ad Hoc для формування повноцінної Mesh-мережі.

Сучасна мережа передачі даних, в тому числі стільникова, повинна зберігати працездатність в екстремальних умовах, навіть якщо одна з ключових частин інфраструктури мережі пошкоджена або повністю зруйнована. Для забезпечення безперервного доступу користувачів до мережі (в цьому випадку, смартфонів через Wi-Fi) і навіть для взаємодії між спеціалістами технічного

обслуговування може бути сформована абсолютно нова мережа, яка здатна на необхідний і достатньо довгий проміжок часу замінити стару та забезпечити безперервну роботу.

Проведено безліч досліджень з метою забезпечення покриття в екстремальних умовах за допомогою високо розташованих платформ HAP (Hight Altitude Platform) систем, таких як “Loon” – проект компанії Google, які привернули увагу до цієї гарячої теми за допомогою своїх HAP на основі надувних кульок [12, 13]. На противагу Google, Facebook також інвестували багато ресурсів у свій власний проект з HAP системами.

Головна ідея розглянутої мережі полягає у можливості, при необхідності, простого та швидкого створення надійної інфраструктури передачі даних з відносно низькою вартістю та простотою технічного обслуговування та її імплементування. Тобто, від команди технічних експертів необхідним буде тільки ввімкнути БпЛА в потрібному місці та вони автоматично займуть необхідно висоту в залежності від рельєфу, формуючи Mesh-мережу один з одним.

Ситуація із розгортання мереж такого типу широко була описана під різними кутами зору. В [14] презентували ідею використання LAP для покриття певної місцевості. Ідея базується на використанні надувних кульок , як і більшість авторів, які працювали над цим питанням, та технологію Wi-Fi для передачі даних. В [15] розширили цю ідею, використовуючи БпЛА з Wi-Fi для формування гнучкої мережевої інфраструктури над ізольованою місцевістю. Проте вони використовували для цих цілей рухомі БпЛА на низькій висоті, які повинні бути близько до клієнтів, для того щоб приймати дані від них. При цьому висота розташування БпЛА досягала декількох сотень метрів, які утворювали між собою Mesh-мережу.

4.5.2 Обґрунтування вибору компонентів мережі

Для досліджень було вибрано технологію стандарту IEEE 802.11s [16] тому, що вона є найбільш універсальною та підтримується мережевими

симуляторами, що дасть можливість перевірити її на життєздатність. В даному розділі вибір компонентів відбувався із урахуванням того, що БпЛА окрім задач відеоспостереження, описаного в минулому розділі, мають забезпечити надійну зону мережевого покриття для клієнтів.

Основними перевагами пропонованого стандарту є надійність і автоматичне формування таблиці маршрутизації, яка використовується, щоб вибрати найкращий шлях. Ці особливості стандарту дозволяють існуючим вузлам оновлювати новий шлях кожен раз, коли вузол додається або вилучається з мережі, що забезпечує високу живучість мережі. Точка доступу та користувач можуть самостійно визначати найкоротший шлях, що також спрощує процес переходу користувача від однієї точки доступу Mesh-мережі до іншої. Також ця технологія є досить енергоефективною, по відношенню до інших.

Однієї з особливостей стандартом IEEE 802.11s є чітке розділення на два типи безпровідних вузлів: перший – це Mesh-маршрутизатори, другий – Mesh-клієнти. Mesh-клієнти не можуть виконувати функцій ретрансляторів, як в повноцінній Ad hoc мережі, що з одного боку знижує її покриття, але довжина шляху негативно впливає на пропускну мережі та враховуючи, що потужність передавача клієнських вузлів невисока і точки доступу повністю покривають всю задану місцевість, це не є ефективним. Для цього сценарію для надання кращої якості для користувачів в приміщенні, можливо розширювати нашу мережеву 3D інфраструктуру наземними Mesh-точками доступу на базі звичайних маршрутизаторів користувачів, оскільки це повністю стандартизовано і підтримується більшістю вендорами.

Питання живлення точок доступу повністю вирішено використанням нових типів БпЛА. Основна перевага таких БпЛА у тому, що він може живитись від БС і може перебувати у повітрі на висоті до 300 м до 2 тижнів повністю автономно, що є важливим аргументом у випадку надзвичайних ситуацій. Також БпЛА оснащений запасною батареєю на випадок короткочасних перебоїв живлення від БС. Слід зазначити, що характеристики відповідають всім погоднім

вимогам, накладеним американськими військовими стандартами та має високу стійкість до вітру.

Приклад з PARC БпЛА із живленням від БС, що знаходиться на землі, наведений на рис. 4.10.

Рис. 4.10 PARC БпЛА із живленням від БС, що знаходиться на землі

4.5.3 Вибір оптимального протоколу маршрутизації

Для вибору оптимальних маршрутів в мережі використовуються різні критерії (метрики). Метрики можуть включати в себе таку інформацію, як довжина шляху (кількість кроків), надійність, затримка, пропускна здатність, завантаження, вартість передачі трафіку тощо. Найбільш розповсюдженою метрикою є довжина шляху. Деякі протоколи маршрутизації дозволяють адміністратору мережі надавати каналу (шлях довжиною в один крок) довільну довжину. При цьому довжина шляху – це сума довжин каналів, через які пролягає шлях від джерела (відправника) до адресата (одержувачу). Інші протоколи визначають число кроків (hop) – скільки мережевих пристроїв

(наприклад, маршрутизаторів) повинен пройти пакет на своєму шляху до одержувача.

Ще один критерій вибору оптимальних маршрутів – надійність. Під метрикою «надійність» зазвичай оцінюють по частці втрат пакетів в кожному з каналів. Деякі канали розриваються частіше, ніж інші, але відновлюються простіше і швидше після помилки в роботі мережі. Будь які фактори надійності можуть враховуватися при отриманні чисельного значення цієї метрики.

Інша популярна метрика – затримка, тобто час, необхідний для доставки пакета від відправника до одержувача. Затримка залежить від багатьох факторів, включаючи пропускну здатність каналів, черги в портах пристроїв на шляху пакета, завантаження мережі у всіх проміжних каналах, а також фізична відстань, яку треба подолати. Пропускна здатність також часто використовується як критерій вибору шляху. Під нею мається на увазі обсяг даних, який може бути переданий по мережі в одиницю часу. Метрика, безпосередньо пов'язана з пропускною здатністю, – це завантаження, яка відображає ступінь зайнятості мережевих ресурсів, таких як канали і маршрутизатори. Слід зазначити, що постійний аналіз цих показників сам по собі може використовувати велику частину ресурсів мережевого обладнання.

Стандарт IEEE 802.11s вимагає, щоб всі пристрої підтримували метрику часу передачі в каналі ALM (Airtime Link Metric). Ця обов’язкова метрика необхідна для сумісності пристроїв і вона задається виразом

$$c_a = \left(O + {B_t}/{r} \right) \Big/ \left(1 - e_f \right), \tag{4.1}$$

де O – накладні витрати доступу до каналу, які включають у себе заголовки пакетів, кадри протоколів доступу тощо; B_t – число бітів в тестовому пакеті (8192); r – швидкість передачі даних в каналі (Мбіт/с); e_f – ймовірність виникнення помилки (вимірюється експериментально на пакетах довжиною B_t). O і B_t – константи.

Ця метрика є оцінкою часу передачі (у секундах) пробного пакета довжиною B_t з урахуванням можливих ретрансляцій при втратах в каналі. Спосіб

визначення параметрів r і e_f у стандарті не наводиться, однак можна припустити, що для цього повинна використовуватися періодична розсилка пробних пакетів довжиною 8192 біт.

В основі методу вибору шляху для передачі даних в стандарте IEEE 802.11s лежить механізм профілів. Цей механізм забезпечує сумісність пристроїв від різних виробників, які можуть підтримувати як стандартизовані механізми, так і власні. Профіль – це запис виду <Ідентифікатор профілю><Ідентифікатор протоколу маршрутизації><Ідентифікатор метрики протоколу маршрутизації>. Пристрій може підтримувати кілька профілів роботи, але одночасно лише один з них може бути активним. Обов'язковий для реалізації профіль використовує протокол HWMP і метрику часу передачі ALM.

4.5.4 Гібридний бездротовий Mesh-протокол маршрутизації HWMP

Гібридний протокол маршрутизації HWMP використовує стандартний набір службових пакетів, правила їх створення і обробки, на кшталт добре відомого протоколу дистанційно-векторної маршрутизації за запитом AODV (Ad hoc On Demand Distance Vector). Однак HWMP адаптований для роботи з адресами MAC-рівня і метриками шляхів. Гібридним він названий тому, що об'єднує в собі два режими побудови шляхів, які можуть бути використані як окремо, так і одночасно в одній мережі:

реактивний режим – побудова маршрутних таблиць в узлах mesh-мережі безпосередньо перед передачею даних (за запитом);

проактивний режим – регулярна процедура оновлення інформації в маршрутних таблиць вузлів всій мережі.

Процедуру ініціює кореневий вузол, в результаті на мережі будується граф (дерево) шляхів з вершиною в кореневому сайті.

Прямим конкурентом є оптимізований протокол стану каналу для бездротової мережі RA-OLSR (Optimized Link State Routing Protocol). Протокол заснований на понятті багатоточкової естафети MPR (Multi Point Relay). Кожен вузол мережі m вибирає кілька вузлів з числа своїх сусідів (тобто вузлів, з якими

у нього встановлено з'єднання). В результаті в мережі формується набір вузлів MPR(*m*). Причому він формується так, що всі вузли, що знаходяться в сфері з радіусом 2 кроки від вузла *m*, мають симетричні канали з MPR(*m*). Це означає, що вузли MPR пов'язані з усіма вузлами в сфері з радіусом 2 кроку. MPR вибираються кожен раз, коли виявляється зміна в сфері з радіусом 1 або 2.

Протокол RA-OLSR ефективний для великих і щільних мобільних мереж і при великій кількості точок доступу по своїм параметрам, перевершує протокол HWMP. Однак він викликав багато зауважень, здебільшого вказують на неточності його опису тексті стандарту і є опціональним. З іншого боку, протокол маршрутизації HWMP обов'язковий для всього стандарту IEEE 802.11s як протокол за замовчуванням. Отже, оскільки наша мережа є відносно невеликою, опціонально може включати точки доступу користувачів при їх бажанні, як одну із вузлів мережі, де важлива простота налаштування, то вибір протоколу HWMP є більш доцільним.

Схематична модель мережі за стандартом IEEE 802.11s наведена на рис. 4.11, а приклад тимчасової мережі на базі БпЛА [17] – на рис. 4.12.

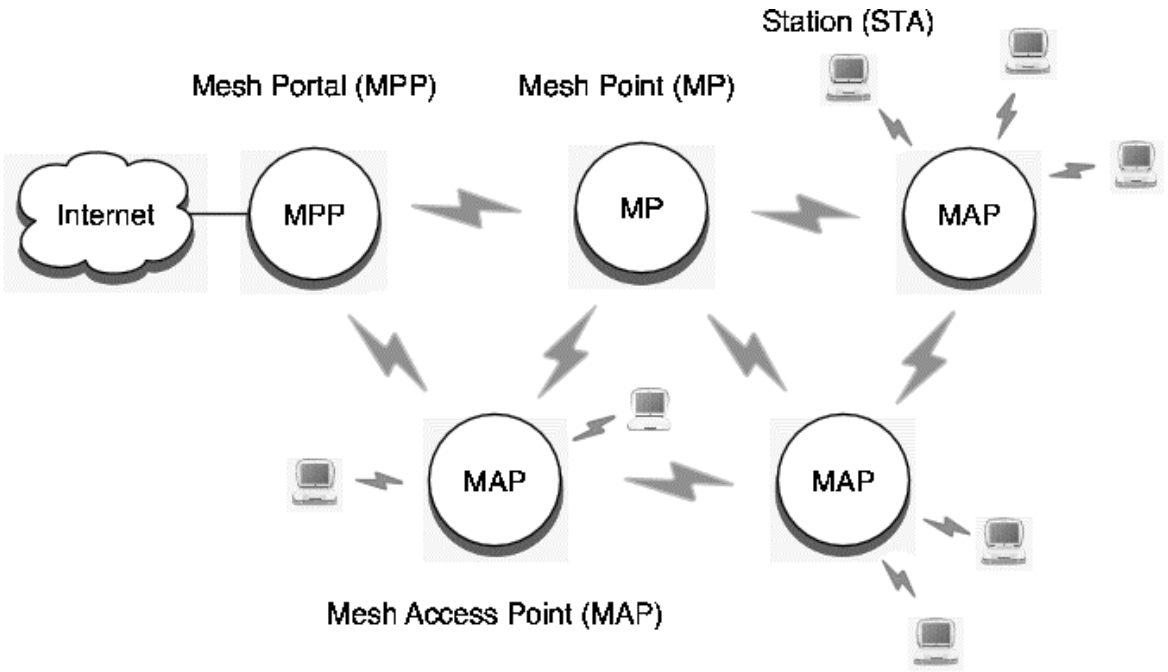


Рис. 4.11 Схематична модель мережі за стандартом IEEE 802.11s

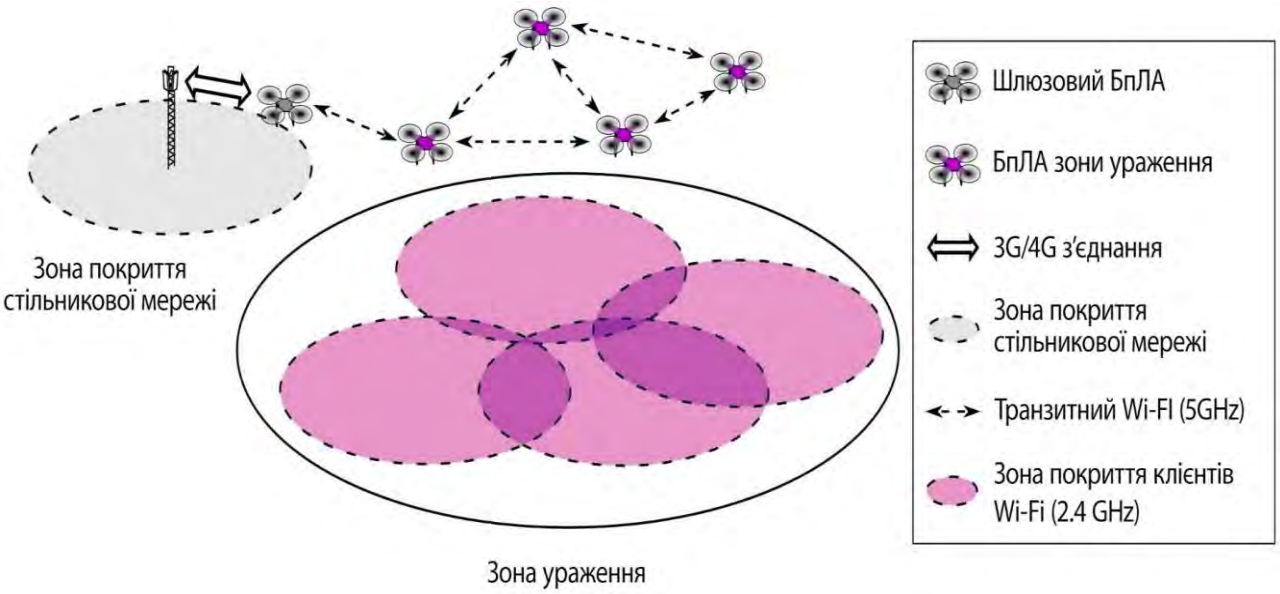


Рис. 4.12 Приклад тимчасової мережі на базі БпЛА

4.5.5 Вибір архітектури з урахуванням функціональної стійкості мережі

LAP концепція, побудована на основі дронів, що оснащені двоканальним Wi-Fi, які формують між собою мережу IEEE 802.11s з використанням 5 ГГц частотного діапазону зі смугою пропускання 40 МГц. Зона покриття формується із точок доступу AP на частоті 2,4 ГГц і полосою пропускання 20 МГц. Перевагою двоканальних точок доступу є розміщення транспортних і користувацьких каналів на різні частотні спектри, що дозволяє їм працювати повністю незаважаючи один одному і уникаючи інтерференції.

З метою з'єднання мережі із глобальною мережею, деякі (один) БпЛА оснащені модемами стільникових мереж третього (3G) або четвертого покоління (4G). Вони називаються шлюзовими і повинні розташовуватись між зоною ураження та працюючою стільниковою мережею. Також треба напряму підключити шлюзовий БпЛА через Ethernet кабель і тим самим забезпечити достатню пропускну здатність. Якщо є можливість підключити кожну із точок доступу на пряму, то і використання WMN є недоцільним.

Схематична структура роботи GW наведена на рис. 4.13.

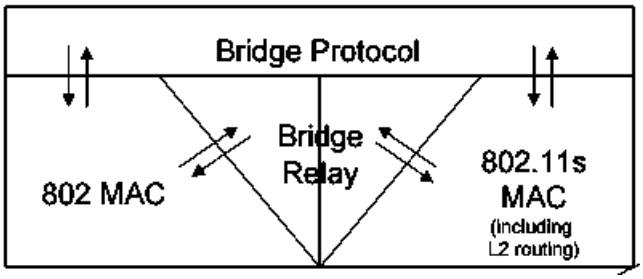


Рис. 4.13 Схематична структура роботи GW

Однак, оскільки моделюється ситуація, при якій телекомунікаційна інфраструктура повністю пошкоджена і немає іншого доступу, окрім як стільникова мережа мобільного оператора, яка нажаль, не може покривати ушкоджену зону і в тому випадку WMN, а саме можливість багатохопового ретранслювання є виходом для реалізації цього сценарію.

Спираючись на результати моделювання покриття цієї мережі, необхідна кількість БпЛА для задоволення телекомунікаційних потреб користувача на прийнятному рівні дорівнює 5x5 БпЛА. Можливо забезпечити і кращу якість покриття використовуючи більше точок доступу та більше шлюзових БпЛА, що підвищити пропускну мережі. Однак оскільки завжди шукається компроміс між ціною та якістю, кількість 5x5 БпЛА буде цілком достатньо, з яких хоча б один (в нашому випадку краще за все 4 для досягнення оптимальної функціональної стійкості та пропускної) повинен виконувати функції, як і шлюзу так і точки доступу.

До основних ознак функціональної стійкості мережі можна віднести:

1. Структура мережі є функціонально стійкою, якщо граф структури однокомпонентний та немає мостів і вузлів з’єднання. Обернене визначення говорить про функціональну нестійкість структури.

2. Структура мережі є функціонально нестійкою, якщо її граф багатокомпонентний і незв’язний.

Таким чином, за зовнішнім видом графа (рис. 4.14), а саме за числом компонентів, наявності мостів і вузлів з’єднання графа можна зробити висновок

про функціональну стійкість структури, тобто про закладені властивості протидіяти відмовам і пошкодженням. Отже, основною вимогою функціональної стійкості структури мережі є зв'язність графа: число вершин зв'язності $\chi(G) \geq 2$, число ребер зв'язності $\lambda(G) \geq 2$. В нашому випадку $\chi(G) = 5$, $\lambda(G) = 13$, що задовільняє критерію функціональної стійкості.

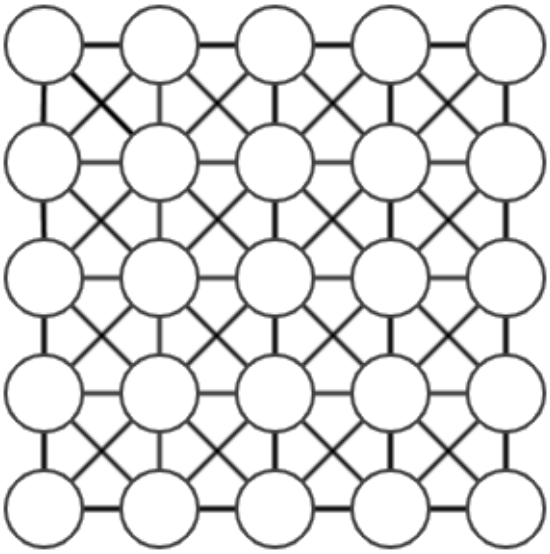


Рис. 4.14 Mesh-мережа 5x5 у вигляді графу

4.6 Моделювання і симуляції функціонально-стійкої Mesh-мережі на основі безпілотних літальних апаратів

В нашому випадку дослідження мобільної Mesh-мережі проводилось на основі симуляції для міської місцевості 6 км×6 км. Моделювання було проведене за допомогою програмного забезпечення WinProp з використання пакету ProMan. Зазвичай використовують для таких цілей базове моделювання з урахуванням втрат у вільному просторі, де відтворюється з'єднання прямої видимості LOS (Line-of-Sight).

Проте для досягнення більш реалістичних результатів для наших цілей було використано детерміновану модель розповсюдження під назвою модель домінантного шляху DPM (Dominant Path Model). DPM проводить розрахунок можливих з допущенням того, що в більшості сценаріїв розповсюдження

радіохвиль існує 1 або 2 основних шляхи, через які проходить 90 % енергії нашого сигналу, які попадають і обробляються приймальним пристроєм. Внаслідок цього, модель домінантного шляху визначає основний шлях між передавачем і приймачем через оцінку прийнятої енергії сигналу. В той же самий час, точність розрахунків DPM аналогічна (а в деяких випадках і краща) моделі трасування променів RTM (Ray Tracing Model), а по обчислювальному часу співрозмірна емпіричній моделі.

Визначення завмирання сигналу в DPM базується на наступному рівняння:

$$L = 20log_{10}\left(4\pi/\lambda\right) + 10nlog_{10}\left(d\right) + \sum_{i=0}^{k} f\left(\varphi,i\right) + \Omega + g_t \qquad (4.2)$$

де d – дистанція між передавачем та приймачем; n – експонента завмирання сигналу; λ – довжина хвилі.

Функція сума втрат індивідуальної взаємодії, через дифракцію для кожної взаємодії і до k з φ, де φ кут між минулим та новою траєкторією поширення.

Рекомендоване значення коефіцієнта завмирання n, залежить від поширення електромагнітних хвиль в просторі та висоти передавача. В [17] провели польові випробування для Wi-Fi на частоті 2,4 ГГц і отримали значення для n в межах 2,54 до 2,76 для з’єднання прямої видимості. В зазначеній роботі обрано значення 2,6 для прямої видимості та 3,5 в протилежному випадку. Максимальна відстань між БпЛА, при якій обмін даними між ними ще можливий, була обчислена для вільного простору і становить 960 м.

Основні технічні параметри БпЛА і їх значення для проведення симуляції були вибрані з табл. 4.2.

Також слід враховувати те, що для Європейських країн, ефективна ізотропна випромінювана потужність EIRP (Equivalent Isotropically Radiated Power) для Wi-Fi обмежена 24 дБм для частот 2,4 ГГц та 36 дБм для 5,8 ГГц. Виходячи із законодавства України будь-яка точка доступу, потужність якої більше 10 дБм потребує спеціального дозволу. В будь якому випадку 10 дБм не задовольняє наших потреб, тому потрібно отримувати цей дозвіл на точки доступу. Для нашого сценарію доцільно і можливо отримати дозвіл на точки

доступу з EIRP 36 дБм, що не є шкідливою для життя людини, оскільки це є нормою для таких країнах, наприклад як США та від цього залежить інфраструктура даної зони.

Таблиця 4.2 Технічні дані БпЛА для симуляції

Параметр	Значення параметра
Wi-Fi частота	2,4 ГГц/5,1 ГГц
Полоса пропускання	20 МГц/40 МГц
Максимальна ефективна ізотропна випромінювана потужність	35 дБм
Висота антени (висота БпЛА)	50–500 м
Модель антени (БпЛА–БпЛА)	HVG-2458-05U
Модель антени (БпЛА–користувач)	NanoStation locoM2
Направленість антени (БпЛА–БпЛА)	5 дБм
Направленість антени (БпЛА–користувач)	8,5 дБм

Виявлено, що зона покриття залежить від висоти зависання БпЛА, як і очікувалось. Найменша зона покриття була в БпЛА на висоті 50 м в 0,14 км^2 (20 дБм. Найбільша ж зона 0,56 км^2 (20 дБм) була досягнута при висоті 500 м. Також було змодельовано, що при збільшенні кількості БпЛА, звичайно ж збільшується і покриття, проте, більше значення висоти зависання з меншою кількістю БпЛА в спромозі надати таку ж якість покриття а то і кращу, як і більша кількість, але з меншим значення висоти зависання.

Як видно з отриманих результатів, якщо використовувати передавачі потужністю 36 дБм, а висота зависання БпЛА буде дорівнювати 300 м, що обумовлене його конструктивними властивостями, то з висотою зависання у 960

м достатньо всього 5x5 БпЛА (точок доступу) для забезпечення якісного покриття.

Залежності покриття від кількості БпЛА, відстані між ними і висоти зависання з потужністю 24 дБм наведено на рис. 4.15, а для з потужністю 36 дБм – на рис. 4.16.

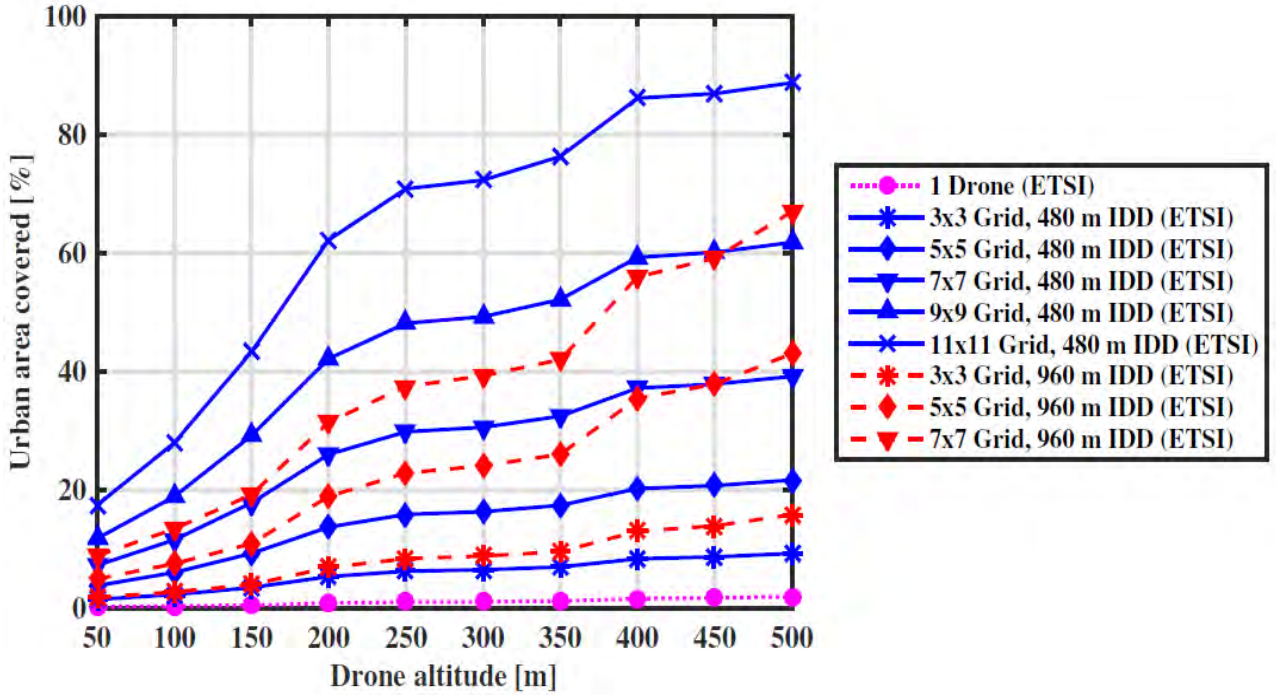


Рис. 4.15 Залежність покриття від кількості БпЛА, відстані між ними та висоти зависання з потужністю 24 дБм.

Для чіткого бачення картини із БпЛА потрібно враховувати затримку між вузлами мережі та забезпечення потреб користувача в повній мірі. Затримка між вузлами мережі включає в себе основні два параметри, до яких відносяться час затримки на обладнанні та час затримки на передачу. Затримка на передачу – це час, за який пакет буде повністю переданий з одного вузла на інший, а затримка на обладнанні – час, за який було виконано обробку, маршрутизацію і т. д. обладнанням перед посилкою певного пакету.

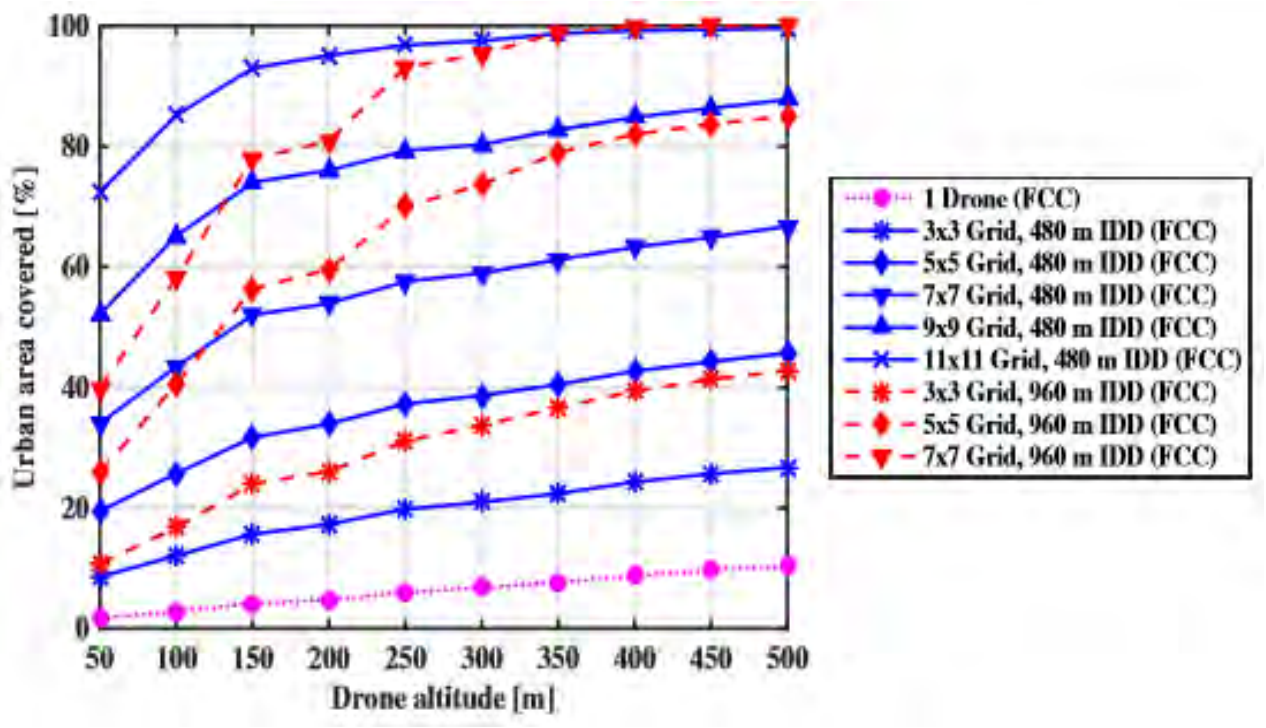


Рис. 4.16 Залежність покриття від кількості БпЛА, відстані між ними та висоти зависання з потужністю 36 дБм.

Схематичне зображення передавання обмін даними між вузлами А та С через проміжний вузол В наведена на рис. 4.17.

Затримку при передачі від вузла А до вузла С через проміжний вузол В можна визначити за виразом:

$$\Delta_{\mathrm{B}}(p) = t_{out,\mathrm{B}}(p) - t_{in,\mathrm{B}}(p), \tag{4.3}$$

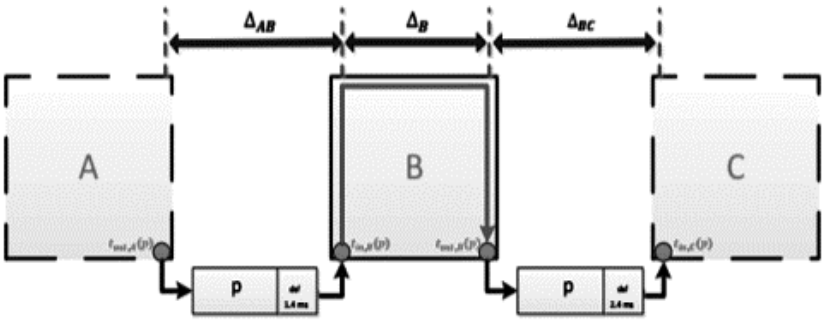


Рис. 4.17 Схематичне зображення передавання обмін даними між вузлами А та С через проміжний вузол В

де $t_{in,\text{B}}(p)$ і $t_{out,\text{B}}(p)$ – час, коли вузол B отримує чи повністю відправляє пакет p відповідно.

Сумарна затримка дорівнює

$$del = \Delta\text{AB} + \Delta\text{B} + \Delta\text{BC} + 2T_{\text{пер}} + 2T_{\text{обл}}, \tag{4.4}$$

Виходячи із специфікації MAC для стандарту IEEE 802.11 при ідеальних умовах в середньому $T_{\text{пер}} = 1{,}4$ мс, а $T_{\text{обл}} = 1{,}0$ мс, то затримка між вузлами A і C, що зображені на рис. 4.17 буде дорівнювати 3,8 мс.

4.7 Особливості використання хмарного провайдера AWS

В свої роботі для підняття БпЛА для симуляції використано хмарний провайдер AWS (Amazon Web Services), яка є дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп’ютерів, який завжди доступний через Інтернет.

Віртуальні комп’ютери AWS мають більшість атрибутів реального комп’ютера, включаючи апаратні пристрої (процесор, відеокарту, локальну та оперативну пам’ять, жорсткий диск або SSD-накопичувач); операційну систему на вибір; мережу; і попередньо встановлені прикладні програми, такі як веб-сервер, база даних, CRM тощо.

Кожна система AWS також віртуалізує консольний ввід/вивід (клавіатура, дисплей і миша), що дозволяє користувачам AWS підключитися до своєї системи AWS за допомогою браузера. Браузер виступає як вікно у віртуальний комп’ютер, дозволяючи користувачу входити в систему, налаштовувати та використовувати свої віртуальні системи так само, як справжній, фізичний комп’ютер. Це дозволяє їм налаштувати систему так, щоб надавати інтернет-орієнтовані сервіси та послуги своїм клієнтам.

Технологія AWS базується на серверних кластерах (фермах), розташованих по всьому світі. Плата за користування базується на комбінації

використання апаратних засобів/ОС/програмного забезпечення/мережевих функцій, вибраних користувачем, а також вимог до доступності, надлишковості, безпеки і додаткових параметрів. Виходячи з того, що користувач потребує і оплачує, він може зарезервувати одну ВМ, кластер ВМ, фізичний (реальний) комп'ютер (Server), призначений для його виняткового використання, або навіть кластер фізичних комп'ютерів (Server Cluster).

Компанія Amazon зобов'язується керувати та оновлювати програмне і апаратне забезпечення для дотримання необхідних стандартів безпеки. AWS працює в багатьох географічних регіонах. Amazon рекламує AWS як спосіб отримання обчислювальної потужності, що масштабується швидше та дешевше, ніж побудова власного фізичного серверного кластеру. Усі послуги оплачуються залежно від використання, однак кожна служба вимірює використання своїм методом.

У 2016 році AWS надавав більш ніж 70 сервісів, що охоплюють широкий спектр, включаючи обчислення та зберігання даних, їхню передачу по мережі, аналітику, мобільні застосунки, інструменти для розробників і т. д. Найпопулярніші з них є Amazon Elastic Compute Cloud (EC2) і Amazon Simple Storage Service (S3). Більшість служб не надаються безпосередньо кінцевим користувачам, але замість цього пропонуються функціональні можливості через API, які розробники можуть використовувати в своїх програмах. Пропозиції Amazon Web Services доступні через Інтернет.

В дослідженні використано безпосередньо сервіс EC2 (Amazon Elastic Compute Cloud) – сервіс IaaS, що надає в користування віртуальні сервери, які контролюються API, основані на гіпервізорі Xen (рис. 4.18). Еквівалентні віддалені сервіси включають Microsoft Azure, Google Compute Engine і Rackspace, а також рішення, які встановлюються на локальні сервери OpenStack або Eucalyptus, веб-сервіс, який надає обчислювальні потужності в хмарі.

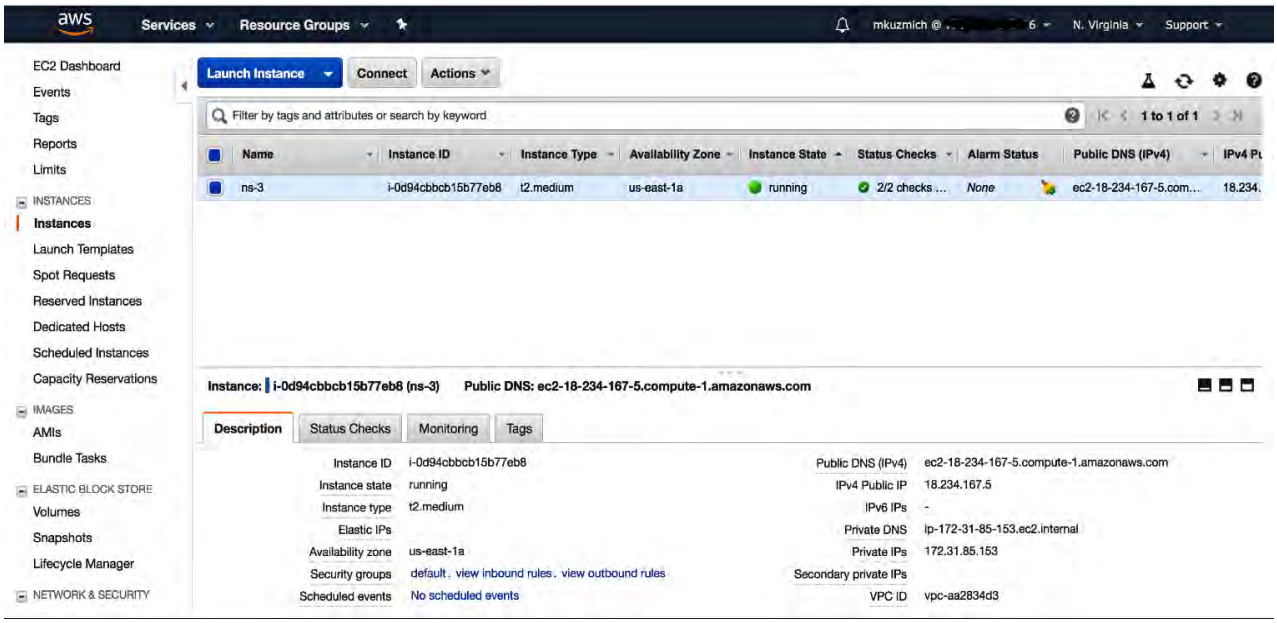


Рис. 4.18 Знімок екрану aws консолі, в якій запущений віртуальний приватний сервер використовучи сервіс EC2

Веб-сервіс входить в інфраструктуру Amazon Web Services. Простий веб-інтерфейс сервісу дозволяє отримати доступ до обчислювальних потужностей і налаштувати ресурси з мінімальними затратами. Він надає користувачам повний контроль над обчислювальними ресурсами, а також доступне середовище для роботи. Служба скорочує час, необхідний для отримання і завантаження нового сервера.

4.8 Особливості використання мережевого симулятора NS-3

NS (Network Simulator) – це назва низки мережевих симуляторів дискретних подій, зокрема, NS-1, NS-2 і NS-3. Всі вони є дискретно-подієвими комп'ютерними мережевими тренажерами, в основному використовується в дослідженнях і викладанні.

Перша версія NS-1 розроблена в Національній лабораторії Л. Берклі в 1995-1997 роках С. МакКенні, С. Флойдом, К. Фолом та іншими дослідниками. Вона була відома як LBNL Network Simulator і була отримана з попереднього симулятора, відомого як REAL С. Кешава. Друга версія NS-2 почалось як варіант

реального мережного симулятора в 1989 році та суттєво еволюціонувала. У 1995 році розвиток NS був здійснений за підтримки компанії Xerox Parc, UCB, і USC/ISI. NS-3 розробили дослідники на чолі з Т. Хендерсоном, у складі Дж. Райлі, С. Флойда і С. Рой. Вони отримали фінансування від американського Національного наукового фонду (NSF), щоб побудувати заміну для NS-2. Ця команда співпрацювала з Planete проект INRIA в Sophia Antipolis, з М. Лакажем як керівником програмного забезпечення, який створив новий проект з відкритим вихідним кодом.

NS-3 – це безкоштовна програма, у відкритому доступі під ліцензією GNU gplv2 – ліцензія на дослідження, розробку і використання. Мета проекту NS-3 – створити відкрите середовище моделювання для дослідження комп'ютерних мереж, які будуть використовуватися всередині наукового співтовариства. Він узгоджений з потребами моделювання сучасних дослідницьких мереж і заохочує внесок співтовариства, експертної оцінки і перевірки програмного забезпечення. Оскільки процес створення мережевого симулятора, який містить достатню кількість високоякісних перевірених, протестованих моделями, які активно підтримуються вимагає багато роботи, проект NS-3 поширює це навантаження на велику групу користувачів і розробників.

Зараз NS розвивається за рахунок підтримки DARPA із SAMAN і NSF з CONSER, і у співробітництві з іншими дослідниками, в т. ч. ACIRI. NS завжди включала істотний код UCB Daedelus і проект CMU Monarch і Sun Microsystems.

У процесі розробки NS-3, було вирішено повністю відмовитися від зворотної сумісності з NS-2. Новий тренажер був написаний з нуля, використовуючи мову програмування C++. Розвиток NS-3 почався в 2006 році. Перший випуск NS-3.1 був зроблений в 2008 році, а потім проект продовжував випускати версії програмного забезпечення кожного кварталу, а останнім часом перейшла на три релізи в рік.

При проведенні досліджень використаний NS-3. Знімок екрану необхідних команд для налаштування NS-3 наведений на рис. 4.19 (вивід команди history).

```
mihail — ubuntu@ip-172-31-85-153: ~ — ssh -i ~/Downloads/test_deploy.pem ubuntu@18.2...
bgsl2 libgsl-dev && sudo apt-get install flex bison libfl-dev && sudo apt-get in
stall tcpdump && sudo apt-get install sqlite sqlite3 libsqlite3-dev && sudo apt-
get install libxml2 libxml2-dev && sudo apt-get install cmake libc6-dev libc6-de
v-i386 libclang-dev && sudo  pip install cxxfilt && sudo apt-get install libgtk2
.0-0 libgtk2.0-dev && sudo apt-get install vtun lxc && sudo apt-get install libb
oost-signals-dev libboost-filesystem-dev
   49  & sudo  pip install cxxfilt && sudo apt-get install libgtk2.0-0 libgtk2.0
-dev && sudo apt-get install vtun lxc && sudo apt-get install libboost-signals-d
ev libboost-filesystem-dev
   50  df -h
   51  cd
   52  cd ns3
   53  ls
   54  ./build.py --enable-examples --enable-tests
   55  ls
   56  ./waf -d debug --enable-examples --enable-tests configure
   57  ls
   58  cd ns-3.27/
   59  ./waf -d debug --enable-examples --enable-tests configure
   60  ./waf
   61  ./test.py
   62  ls
   63  cp examples/wireless/wifi-adhoc.cc scratch/.
   64  ls
```

Рис. 4.19 Знімок екрану необхідних команд для налаштування NS-3

NS-3 побудована з використанням мови C++ і Python для скриптових можливостей. NS-3 бібліотеки обгорнуті Python завдяки pybindgen бібліотеки, яка відправляє розкладені C++ заголовки NS-3 на gccxml і pygccxml, щоб автоматично генерувати відповідну C++ зв’язуючу вставку. Ці автоматично згенеровані файли C++ остаточно компілюються в модуль Python середовища NS-3, щоб дозволити користувачам взаємодіяти з C++ NS-3 моделлю і ядром за допомогою скриптів Python. NS-3 тренажер має вбудований атрибут-системи для управління за замовчуванням і для кожного екземпляра значення для моделювання параметрів.

Знімок екрану структури UDP пакету в mesh-мережі наведено на рис. 4.20, а знімок екрану просимульоваго трафіку в програмі Wireshark – на рис. 4.21.

```
▼ IEEE 802.11 QoS Data, Flags: ......FT
    Type/Subtype: QoS Data (0x0028)
  ▶ Frame Control Field: 0x8803
    .000 0000 0010 1100 = Duration: 44 microseconds
    Receiver address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Transmitter address: 00:00:00_00:00:09 (00:00:00:00:00:09)
    Destination address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Source address: 00:00:00_00:00:11 (00:00:00:00:00:11)
    .... .... .... 0000 = Fragment number: 0
    0000 0110 1010 .... = Sequence number: 106
  ▶ Qos Control: 0x0100
  ▶ Mesh Control Field
▼ Logical-Link Control
  ▶ DSAP: SNAP (0xaa)
  ▶ SSAP: SNAP (0xaa)
  ▶ Control field: U, func=UI (0x03)
    Organization Code: 00:00:00 (Officially Xerox, but
    Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 10.1.1.9, Dst: 10.1.1.1
▼ User Datagram Protocol, Src Port: 49153, Dst Port: 9
    Source Port: 49153
    Destination Port: 9
    Length: 1032
    [Checksum: [missing]]
    [Checksum Status: Not present]
    [Stream index: 0]
▼ Data (1024 bytes)
    Data: 0000000000000000000000000000000000000000000000000...
    [Length: 1024]
```

Рис. 4.20 Знімок екрану структури UDP пакету в mesh-мережі

10265	2.743880	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10266	2.744474	10.1.1.9	10.1.1.1	UDP	1102	49153 → 9 Len=1024
10267	2.744490		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10268	2.744579	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10269	2.744807		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=.....
10270	2.744931	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10271	2.745458	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10272	2.745686		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=.....
10273	2.745940	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10274	2.745984		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10275	2.746297		00:00:00_00:00:11 (00:00:00:00:00:11) (RA)	802.11	14	Acknowledgement, Flags=.....
10276	2.746551	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10277	2.746595		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10278	2.746926		00:00:00_00:00:11 (00:00:00:00:00:11) (RA)	802.11	14	Acknowledgement, Flags=.....
10279	2.747222	10.1.1.9	10.1.1.1	UDP	1102	49153 → 9 Len=1024
10280	2.747238		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10281	2.747511	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10282	2.747556		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10283	2.747869		00:00:00_00:00:11 (00:00:00:00:00:11) (RA)	802.11	14	Acknowledgement, Flags=.....
10284	2.748150	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024
10285	2.748194		00:00:00_00:00:09 (00:00:00:00:00:09) (RA)	802.11	14	Acknowledgement, Flags=.....
10286	2.748237	10.1.1.1	10.1.1.9	UDP	1102	9 → 49153 Len=1024

Рис. 4.21 Знімок екрану просимульоваго трафіку в програмі Wireshark

Графіки оброблених UDP пакетів між двома вузлами мережі в залежності від часу, Conversion = 6739 кбіт/с і Conversion = 6739 кбіт/с наведені відповідно на рис. 4.22 і рис. 4.23.

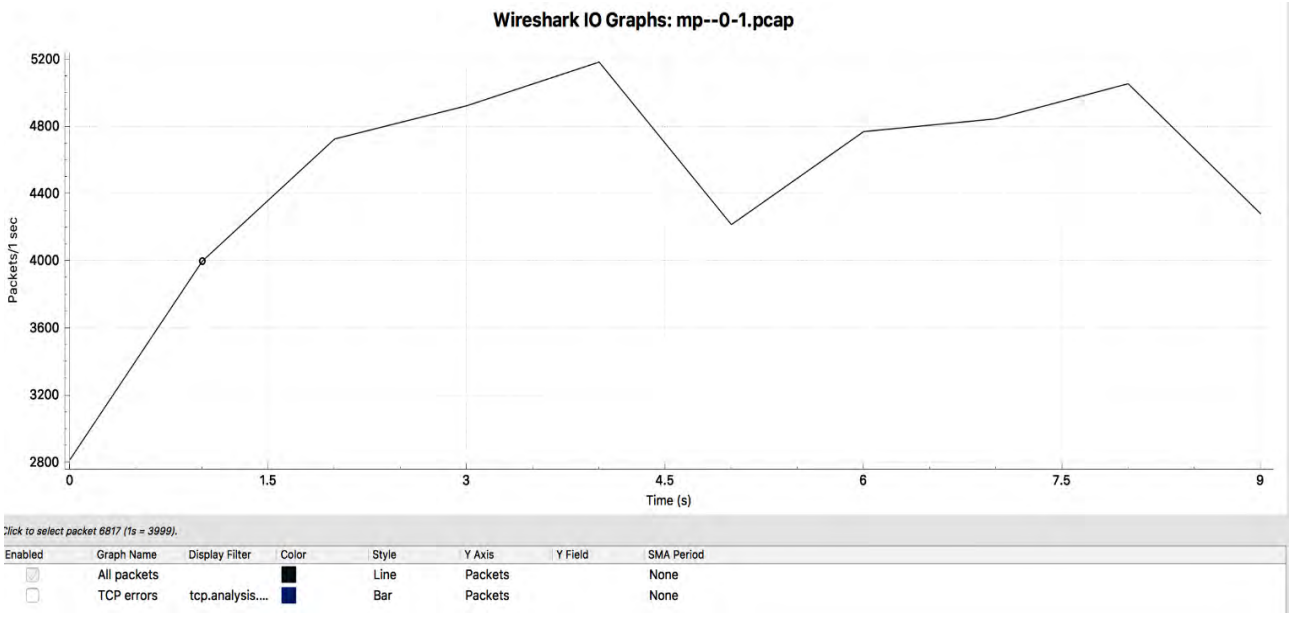


Рис. 4.22 Графік оброблених UDP пакетів, Conversion = 6739 кбіт/с

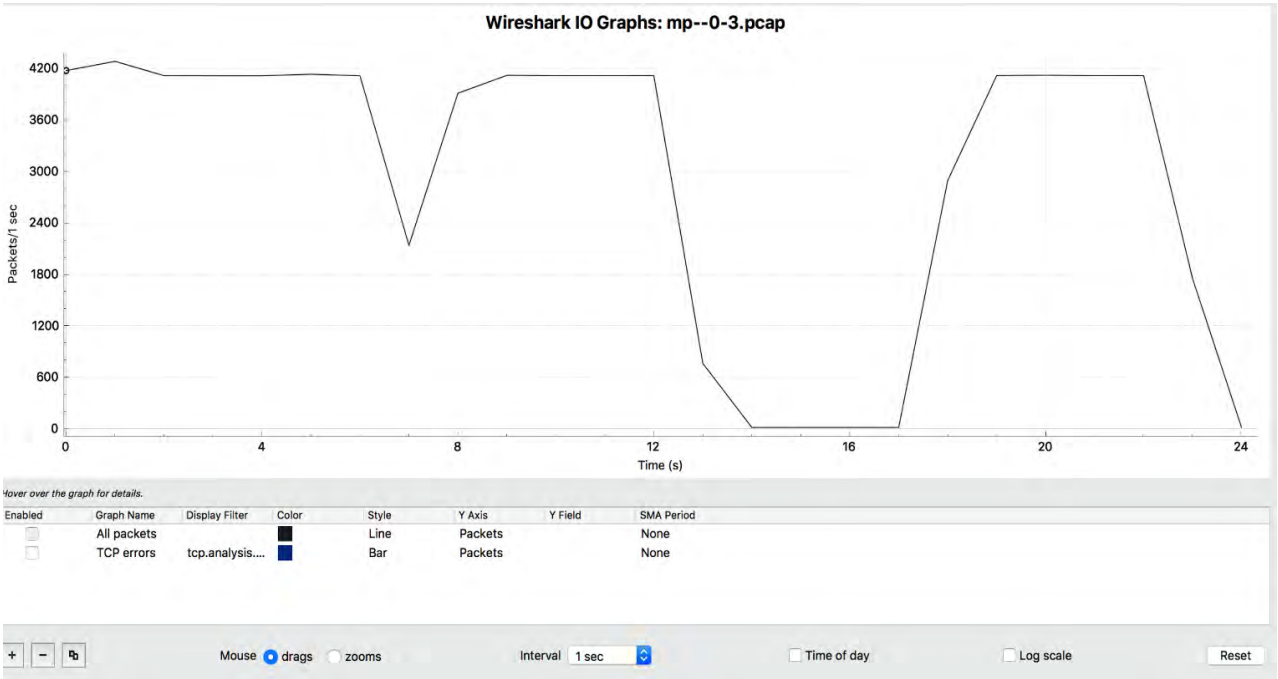


Рис. 4.23 Графік оброблених UDP пакетів, Conversion = 6668 кбіт/с

Симуляцію можна вважати успішною, оскільки як показує графіки UDP вхідних каналів і трейс, трафік ходить по мережі без проблем і стандарт IEEE 802.11s з протоколом маршрутизації HWMP, який є базовим для стандарту IEEE 802.11s може бути використано на практиці. Максимальну та середню пропускну було знайдемо шляхом генерування трафіку 10000 UDP пакетів в секунду (m_packetInterval (0.0001). Max conversion = 5200*8*1024 = 42,5 Мбіт/с. В Додатку А представлена структура інформаційного пакету переданого по Mesh-мережі.

4.9 Висновки до розділу

Розроблено концепцію конвеєра МН для моделі виявлення об’єктів для БпЛА з можливістю постійного донавчання. Це сприяє швидкій адаптації інформаційної системи до змін в навколишньому середовищі та специфіки завдань.

Результатом проведеного дослідження є формування та обґрунтування конвеєра із використанням компонентів платформи Kubeflow. На основі чого можна зробити висновок, що Kubeflow є довершеною платформою для розробки та впровадження програмних продуктів, що базуються на МН. Наявність великого функціонала робить Kubeflow ефективним інструментом розробки та впровадження МН моделей на підприємствах, в тому числі із невеликим штатом інженерів. З іншого боку велика кількість абстракцій в системі вимагає від кінцевого користувача глибоких знань в предметній галузі, коли необхідно поглибитись на один або декілька рівнів нижче для кастоматизації.

Використання Kubernetes як платформи для конвеєру, робить систему портативною, що дає можливість переносити та використовувати різні рішення та моделі МН без прив’язки до конкретного середовища чи платформи. Описана модель буде ефективно працювати як і в хмарному середовищі, так і в парадигмі Edge Computing. На основі проведено аналізу основних дистрибутив Kubernetes, що використовуються для Edge Computing, обґрунтована переваги та доцільність використання k3s для поточної системи. Запропоновано використання Mesh-

мереж для БпЛА, які є розподіленою інформаційною системою на основі дистрибутива Kubernetes k3s та Kubeflow, що працюють в парадигмі Edge Computing. Доведено, що використання Mesh-мереж дозволяє досягти високої функціональної стійкості інформаційної системи БпЛА в умовах ведення радіоелектронної боротьби.

Наведена концепція використання MLOps конвеєру для розв'язання прикладної задачі класифікації об'єктів із відео розвідувальних БпЛА на основі системи мобільного агента, що працюють як повноцінна розподілена система. Запропоноване застосування Mesh-мереж для досягнення високої функціональної стійкості інформаційної системи БпЛА, що робить її використання можливим в умовах ведення радіоелектронної боротьби.

Для вирішення задачі простого та швидкого розгортання мережі передачі даних було запропоновано Mesh-мережу стандарту IEEE 802.11s на основі БпЛА. Було визначено оптимальну кількість точок доступу, їх висоту зависання для покриття території 6х6 км. Також, утворену мережу можна вважати функціонально стійкою, оскільки виконується основна вимога, а саме зв'язність графу мережі. Працездатність мережі було перевірено з використанням мережевого симулятора NS-3, який було встановлено та налаштовано із використанням хмарних технологій від компанії AWS.

4.10 Список використаних джерел

1. D. Kunertova. Drones have boots: Learning from Russia's war in Ukraine. Contemporary Security Policy, vol. 44, 2023, no.4, pp. 576–591.

2. STANAG 4670 Ed. 4 / ATP-3.3.8.1 Ed. A. Minimum training requirements for Unmanned Aircraft Systems (UAS) operators and pilots, NATO, 2019.

3. Кузьміч М. Ю., Гордієнко Т. Б. Упровадження Kubeflow MLOps у розподілені інформаційні системи мобільних агентів із покращеною функціональною стійкістю, Зв'язок, 2023, № 6 (166), с. 28–32.

4. Кузьміч М. Ю., Гордієнко Т. Б. Застосування інструменту Kubeflow для інтеграції машинного навчання і штучного інтелекту в безпілотних літальних

апаратах. Телекомунікаційні та інформаційні технології, 2023, вип. 3 (80), с. 65–78.

5. Mesh Rider® Radios. [Електронний ресурс] – Режим доступу до ресурсу: https://doodlelabs.com/mesh-rider-radios/ (дата звернення 01.11.2023).

6. C100 [Електронний ресурс] – Режим доступу до ресурсу: https://www.pdw.ai/c100 (дата звернення 01.11.2023).

7. Кузьміч М. Ю., Кравчук С. О., "Дослідження мобільної Mesh-мережі безпілотних літальних апаратів з урахуванням затримки між вузлами". Збірник матеріалів Міжнародної науково-технічної конференції "Перспективи телекомунікацій", Київ: Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, 2017, с. 63–65.

8. Кузьміч М. Ю., Павлов А. О., "Інформаційна система ідентифікації користувачів на основі біометричних даних". II Міжнар. наук.-практ. конф. "Системний аналіз та інтелектуальні системи для бізнесу та управління", Київ: ДУТ, 2023, с. 39–43.

9. Кузьміч М. Ю., Костенко В. К., "Інформаційна система управління проектами". II Міжнар. наук.-практ. конф. "Системний аналіз та інтелектуальні системи для бізнесу та управління", Київ: ДУТ, 2023, с. 106–108.

10. Кузьміч М. Ю., Дубовой Д. В., "Розробка інформаційної системи рекомендування ціни вживаного авто". II Міжнар. наук.-практ. конф. "Системний аналіз та інтелектуальні системи для бізнесу та управління", Київ: ДУТ, 2023, с. 194–198.

11. Кузьміч М. Ю., Сміхун О. Ф., "Інформаційна система аналізу зміни стану рослин сільськогосподарських угідь з урахуванням даних їх дистанційного зондування". II Міжнар. наук.-практ. конф. "Системний аналіз та інтелектуальні системи для бізнесу та управління", Київ: ДУТ, 2023, с. 111–116.

12. Y. Xing, F. Hsieh, A. Ghosh and T. S. Rappaport, "High Altitude Platform Stations (HAPS): Architecture and System Performance". 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 2021, pp. 1-6.

13. A. K. Widiawan and R. Tafazolli, “High Altitude Platform Station (HAPS): A Review of NewInfrastructure Development for Future Wireless Communications”. Wireless Personal Communications”, vol. 42, 2007, pp. 387–404.

14. H. Hariyanto, H. Santoso, and A. Widiawan, “Emergency broadband access network using low altitude platform”. In Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICIBME), 2009 International Conference on, Nov 2009, pp. 1–6.

15. N. Uchida et al., “Evaluation of Wireless Network Communication by Autonomous Flight Wireless Nodes for Resilient Networks”. 2014 17th International Conference on Network-Based Information Systems, Salerno, Italy, 2014, pp. 180–185.

16. IEEE 802.11s, “The WLAN Mesh Standard”. IEEE, 2009.

17. J. Sae, S. F. Yunas and J. Lempi, “Coverage Aspects of Temporary LAP Network”. 2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS), 2016, pp. 100–103.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-технічну задачу, що полягає в розробленні методики побудови та використання моделей МН для мобільних агентів, якими виступають БпЛА, за допомогою автоматизованого конвеєра на основі Kubeflow із використанням кращих практик концепції МН та процесів MLOps. Ця задача передбачає необхідність удосконалення моделі підвищення функціональної стійкості розподіленої інформаційної системи БПлА із використанням Mesh-мереж, що є частиною описаного конвеєру.

Найбільш значущі результати роботи полягають у наступному:

1. Проведено аналіз концепції MLOps в розрізі інструментів Kubeflow із використанням низки визначених факторів покращення, зокрема таких як швидкість розробки, зменшення кількості помилок в моделі, часу на пошук проблем тощо. Аналіз показав, що можливості сучасних MLOps рішень можуть сприяти покращенню процесів розробки інформаційних систем МН. Висока портативність Kubernetes та Kubeflow дозволяє розгортати додатки на різних систем, а концепція системи на базі дистрибутивів Kubernetes дає можливість встановлювати їх на мобільних агентах (БпЛА). Зважаючи на зазначене, вперше сформовано концепцію повноцінного інформаційного рішення на базі конвеєру неперервної інтеграції із можливістю постійного донавчання, високого рівня інтеграції та автоматизації з використанням компонентів Kubeflow і платформи Kubernetes. Це дозволило покращити кількісну та якісну складові експериментів, зменшити час, затрачений на його підготовку, та мінімізувати помилки, спричинені людським фактором. Розроблена концепція конвеєра МН для моделі виявлення об’єктів БпЛА з можливістю постійного донавчання сприяє швидкій адаптації інформаційної системи до змін в навколишньому середовищі та специфіки завдань.

2. Запропоновано використання бездротових мереж на кшталт Mesh-мереж для підвищення функціональної стійкості розподіленої інформаційної системи мобільних агентів. Особливу увагу було приділено стандарту IEEE 802.11s на

базі якого і була змодельована Mesh-мережа на основі БпЛА. Вперше спроектовано архітектурну концепцію системи на базі дистрибутива Kubernetes k3s. Це дозволяє ефективно використовувати моделі МН в парадигмі граничних обчислень (Edge Computing) із використанням Mesh-мереж для мобільних агентів (БпЛА), що формують собою функціонально стійку розподілену інформаційну систему. Працездатність мережі перевірено з використанням мережевого симулятора NS-3, який було встановлено та налаштовано із використанням хмарних технологій від компанії AWS.

3. Удосконалено модель підвищення функціональної стійкості розподіленої інформаційної системи БПлА з використанням бездротової Mesh-мережі передачі даних, що дозволить протистояти радіоелектронним та іншим перешкодам і працювати в автономному або напів-автономному режимі.

4. Запропоновано архітектурні рішення інформаційних систем та метод можуть бути використані дослідно-конструкторськими організаціями та державними структурами Збройних сил України для реалізації концепції «Аналітична система підтримки прийняття рішень для БпЛА», що показано через функціональні моделі процесів (AS, TO-BE, SHOULD-BE).

5. Сучасні MLOps рішень на базі інструментів Kubeflow для створення та використання навчальних моделей МН впроваджено в дисципліну «Системи підтримки прийняття рішень» в навчальний процес студентів галузі знань 12 «Інформаційні технології» Державного університету інформаційно-комунікаційних технологій. Високий рівень автоматизації та інтеграції компонентів дозволив створити та запустити тестову модель за відносно короткий час, але без зниження якості її роботи та надійності.

ДОДАТКИ

ДОДАТОК А
ЛІСТИНГИ РОЗРАХУНКОВО-ЕКСПЕРИМЕНТАЛЬНИХ МОДЕЛЕЙ

А1. Мережевий пакету в програмі Wireshark просимульованої мережі

```
/* https://github.com/nsnam/ns-3-dev-
git/blob/master/src/mesh/examples/mesh.cc
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2008,2009 IITP RAS
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 * Author: Kirill Andreev <andreev@iitp.ru>
 *
 *
 * By default this script creates m_xSize * m_ySize square grid topology with
 * IEEE802.11s stack installed at each node with peering management
```

```
 * and HWMP protocol.
 * The side of the square cell is defined by m_step parameter.
 * When topology is created, UDP ping is installed to opposite corners
 * by diagonals. packet size of the UDP ping and interval between two
 * successive packets is configurable.
 *
 *  m_xSize * step
 *  |<--------->|
 *   step
 *  |<--->|
 *  * --- * --- * <---Ping sink  _
 * |       |      |              ^
 * |       |      |              |
 *  * --- * --- * m_ySize * step |
 * |       |      |              |
 * |       |      |              |
 *  * --- * --- *                _
 *  ^ Ping source
 *
 *  See also MeshTest::Configure to read more about configurable
 *  parameters.
 */

#include <iostream>
#include <sstream>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
```

```
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/mesh-helper.h"
#include "ns3/yans-wifi-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TestMeshScript");

/**
 * \ingroup mesh
 * \brief MeshTest class
 */
class MeshTest
{
public:
  /// Init test
  MeshTest ();
  /**
   * Configure test from command line arguments
   *
   * \param argc command line argument count
   * \param argv command line arguments
   */
  void Configure (int argc, char ** argv);
  /**
   * Run test
   * \returns the test status
   */
  int Run ();
```

```
private:
  int       m_xSize; ///< X size
  int       m_ySize; ///< Y size
  double    m_step; ///< step
  double    m_randomStart; ///< random start
  double    m_totalTime; ///< total time
  double    m_packetInterval; ///< packet interval
  uint16_t  m_packetSize; ///< packet size
  uint32_t  m_nIfaces; ///< number interfaces
  bool      m_chan; ///< channel
  bool      m_pcap; ///< PCAP
  bool      m_ascii; ///< ASCII
  std::string m_stack; ///< stack
  std::string m_root; ///< root
  /// List of network nodes
  NodeContainer nodes;
  /// List of all mesh point devices
  NetDeviceContainer meshDevices;
  /// Addresses of interfaces:
  Ipv4InterfaceContainer interfaces;
  /// MeshHelper. Report is not static methods
  MeshHelper mesh;
private:
  /// Create nodes and setup their mobility
  void CreateNodes ();
  /// Install internet m_stack on nodes
  void InstallInternetStack ();
  /// Install applications
  void InstallApplication ();
  /// Print mesh devices diagnostics
```

```
  void Report ();
};
MeshTest::MeshTest () :
  m_xSize (5),
  m_ySize (5),
  m_step (100.0),
  m_randomStart (0.1),
  m_totalTime (25.0),
  m_packetInterval (0.0001),
  m_packetSize (1024),
  m_nIfaces (2),
  m_chan (true),
  m_pcap (false),
  m_ascii (false),
  m_stack ("ns3::Dot11sStack"),
  m_root ("ff:ff:ff:ff:ff:ff")
{
}
void
MeshTest::Configure (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.AddValue ("x-size", "Number of nodes in a row grid", m_xSize);
  cmd.AddValue ("y-size", "Number of rows in a grid", m_ySize);
  cmd.AddValue ("step",   "Size of edge in our grid (meters)", m_step);
  // Avoid starting all mesh nodes at the same time (beacons may collide)
  cmd.AddValue ("start",  "Maximum random start delay for beacon jitter
(sec)", m_randomStart);
  cmd.AddValue ("time",  "Simulation time (sec)", m_totalTime);
```

```
    cmd.AddValue ("packet-interval",  "Interval between packets in UDP ping
(sec)", m_packetInterval);
    cmd.AddValue ("packet-size",  "Size of packets in UDP ping (bytes)",
m_packetSize);
    cmd.AddValue ("interfaces", "Number of radio interfaces used by each mesh
point", m_nIfaces);
    cmd.AddValue ("channels",   "Use different frequency channels for different
interfac-es", m_chan);
    cmd.AddValue ("pcap",   "Enable PCAP traces on interfaces", m_pcap);
    cmd.AddValue ("ascii",   "Enable Ascii traces on interfaces", m_ascii);
    cmd.AddValue ("stack",  "Type of protocol stack. ns3::Dot11sStack by
default", m_stack);
    cmd.AddValue ("root", "Mac address of root mesh point in HWMP",
m_root);

    cmd.Parse (argc, argv);
    NS_LOG_DEBUG ("Grid:" << m_xSize << "*" << m_ySize);
    NS_LOG_DEBUG ("Simulation time: " << m_totalTime << " s");
    if (m_ascii)
      {
        PacketMetadata::Enable ();
      }
  }
  void
  MeshTest::CreateNodes ()
  {
    /*
     * Create m_ySize*m_xSize stations to form a grid topology
     */
    nodes.Create (m_ySize*m_xSize);
```

```
// Configure YansWifiChannel
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
/*
 * Create mesh helper and set stack installer to it
 * Stack installer creates all needed protocols and install them to
 * mesh point device
 */
mesh = MeshHelper::Default ();
if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
  {
    mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue
(Mac48Address (m_root.c_str ())));
  }
else
  {
    //If root is not set, we do not use "Root" attribute, because it
    //is specified only for 11s
    mesh.SetStackInstaller (m_stack);
  }
if (m_chan)
  {
    mesh.SetSpreadInterfaceChannels (MeshHelper::SPREAD_CHANNELS);
  }
else
  {
    mesh.SetSpreadInterfaceChannels (MeshHelper::ZERO_CHANNEL);
  }
mesh.SetMacType ("RandomStart", TimeValue (Seconds (m_randomStart)));
```

```
  // Set number of interfaces - default is single-interface mesh point
  mesh.SetNumberOfInterfaces (m_nIfaces);
  // Install protocols and return container if MeshPointDevices
  meshDevices = mesh.Install (wifiPhy, nodes);
  // Setup mobility - static grid topology
  MobilityHelper mobility;
  mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (m_step),
                                 "DeltaY", DoubleValue (m_step),
                                 "GridWidth", UintegerValue (m_xSize),
                                 "LayoutType", StringValue ("RowFirst"));
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);
  if (m_pcap)
    wifiPhy.EnablePcapAll (std::string ("mp-"));
  if (m_ascii)
    {
      AsciiTraceHelper ascii;
      wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("mesh.tr"));
    }
}
void
MeshTest::InstallInternetStack ()
{
  InternetStackHelper internetStack;
  internetStack.Install (nodes);
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
```

```
  interfaces = address.Assign (meshDevices);
}
void
MeshTest::InstallApplication ()
{
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (Seconds (m_totalTime));
  UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue
((uint32_t)(m_totalTime*(1/m_packetInterval))));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds
(m_packetInterval)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (m_packetSize));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get
(m_xSize*m_ySize-1));
  clientApps.Start (Seconds (0.0));
  clientApps.Stop (Seconds (m_totalTime));
}
int
MeshTest::Run ()
{
  CreateNodes ();
  InstallInternetStack ();
  InstallApplication ();
  Simulator::Schedule (Seconds (m_totalTime), &MeshTest::Report, this);
  Simulator::Stop (Seconds (m_totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
```

```
  return 0;
}
void
MeshTest::Report ()
{
  unsigned n (0);
  for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i !=
meshDevices.End (); ++i, ++n)
    {
      std::ostringstream os;
      os << "mp-report-" << n << ".xml";
      std::cerr << "Printing mesh point device #" << n << " diagnostics to " <<
os.str () << "\n";
      std::ofstream of;
      of.open (os.str ().c_str ());
      if (!of.is_open ())
        {
          std::cerr << "Error: Can't open file " << os.str () << "\n";
          return;
        }
      mesh.Report (*i, of);
      of.close ();
    }
}
int
main (int argc, char *argv[])
{
  MeshTest t;
  t.Configure (argc, argv);
  return t.Run ();
```

}

А2. Код симуляції Mesh-мережі 5x5 стандарту IEEE 802.11s

Frame 44658: 1102 bytes on wire (8816 bits), 1102 bytes captured (8816 bits)

IEEE 802.11 QoS Data, Flags: .....FT

Type/Subtype: QoS Data (0x0028)

Frame Control Field: 0x8803

.000 0000 0011 0000 = Duration: 48 microseconds

Receiver address: 00:00:00_00:00:09 (00:00:00:00:00:09)

Transmitter address: 00:00:00_00:00:01 (00:00:00:00:00:01)

Destination address: 00:00:00_00:00:11 (00:00:00:00:00:11)

Source address: 00:00:00_00:00:01 (00:00:00:00:00:01)

.... 0000 = Fragment number: 0

0111 0110 0110 .... = Sequence number: 1894

Qos Control: 0x0100

Mesh Control Field

Mesh Flags: 0x00

Mesh TTL: 0x20

Sequence Number: 0x00000000

Logical-Link Control

DSAP: SNAP (0xaa)

SSAP: SNAP (0xaa)

Control field: U, func=UI (0x03)

Organization Code: 00:00:00 (Officially Xerox, but

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.1.9

0100 .... = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 1052

Identification: 0x17a0 (6048)

Flags: 0x0000

Time to live: 64

Protocol: UDP (17)

Header checksum: 0x0000 [validation disabled]

[Header checksum status: Unverified]

Source: 10.1.1.1

Destination: 10.1.1.9

User Datagram Protocol, Src Port: 9, Dst Port: 49153

Source Port: 9

Destination Port: 49153

Length: 1032

[Checksum: [missing]]

[Checksum Status: Not present]

[Stream index: 0]

Data (1024 bytes)

Data: 0000000000000000000000000000000000000000000000000...

[Length: 1024]

ДОДАТОК Б
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. M. Kuzmich, T. Gordiyenko, “Application of Kubeflow as a universal approach for the development and implementation of artificial intelligence systems”, ARPN Journal of Engineering and Applied Sciences, vol. 18, 2023, no. 20, pp. 2311–2320 (**Scopus**).

2. Кузьміч М. Ю., Гордієнко Т. Б., “Застосування інструменту Kubeflow для інтеграції машинного навчання і штучного інтелекту в безпілотних літальних апаратах”. Телекомунікаційні та інформаційні технології, 2023, вип. 3 (80), с. 65–78.

3. Кузьміч М. Ю., Гордієнко Т. Б., “Імплементація та автоматизація розгортання WebRTC застосунків в Cloud Native оточенні”. Телекомунікаційні та інформаційні технології, 2021, вип. 3 (72), с. 54–62.

4. Кузьміч М. Ю., Гордієнко Т. Б., “Упровадження Kubeflow MLOps у розподілені інформаційні системи мобільних агентів із покращеною функціональною стійкістю”. Зв’язок, 2023, № 6 (166), с. 28–32.

Наукові праці, які засвідчують апробацію матеріалів дисертації

5. Kuzmich M. Ю., Zolotuhina O. A., “Analysis of challenges faced by implementation WebRTC SFU gateway in cluster computing environment”. Збірник матеріалів Науково-технічної конференції “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”, Київ: ДУТ, 2020, с. 58–60.

6. Кузьміч М. Ю., “Особливості імплементація WebRTC застосунків в cloud native оточенні”. Збірник матеріалів науково-практичної конференцію “Проблеми комп’ютерної інженерії”, Київ: ДУТ, 2020, с. 23–25.

7. Кузьміч М. Ю., Кравчук С. О., “Дослідження мобільної Mesh-мережі безпілотних літальних апаратів з урахуванням затримки між вузлами”. Збірник

матеріалів Міжнародної науково-технічної конференції “Перспективи телекомунікацій”, Київ: Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, 2017, с. 63–65.

Наукові праці, які додатково відображають наукові результати дисертації

8. Кузьміч М. Ю., Павлов А. О., “Інформаційна система ідентифікації користувачів на основі біометричних даних”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 39-43.

9. Кузьміч М. Ю., Костенко В. К., “Інформаційна система управління проектами”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 106–108.

10. Кузьміч М. Ю., Дубовой Д. В., “Розробка інформаційної системи рекомендування ціни вживаного авто”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 194–198.

11. Кузьміч М. Ю., Сміхун О. Ф., “Інформаційна система аналізу зміни стану рослин сільськогосподарських угідь з урахуванням даних їх дистанційного зондування”. II Міжнар. наук.-практ. конф. “Системний аналіз та інтелектуальні системи для бізнесу та управління”, Київ: ДУТ, 2023, с. 111–116.

ДОДАТОК В

АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Документ підписано у сервісі Вчасно (початок)
dis_kuzmich.pdf

Документ підписано у сервісі Вчасно (продовження)
dis_kuzmich.pdf

Документ відправлено: 10:46 04.01.2024

Власник документу

Електронний підпис
10:46 04.01.2024
Ідентифікаційний код: 3496012035
Кузьміч Михайло Юрійович
Власник ключа: Кузьміч Михайло Юрійович
Час перевірки КЕП/ЕЦП: 10:46 04.01.2024
Статус перевірки сертифікату: Сертифікат діє
Серійний номер: 3ED5083160DBC59B04000000EF7111007A95F100
Тип підпису: кваліфікований