https://doi.org/10.31891/2219-9365-2026-86-30
УДК 004.75:004.45:621.39

КОРЕЦЬКИЙ Олександр
Державний університет інформаційно-комунікаційних технологій
https://orcid.org/0009-0001-4809-7556
e-mail: okoretsky@gmail.com

АРХІТЕКТУРНА МОДЕЛЬ МЕТОДУ ПОБУДОВИ ПРОГРАМНОЇ ПЛАТФОРМИ МІКРОСЕРВІСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В EDGE/FOG-СЕРЕДОВИЩІ

У статті запропоновано архітектурну модель програмної платформи методу побудови програмної платформи мікросервісного програмного забезпечення в Edge/Fog -середовищі. На відміну від статичних підходів до розміщення сервісів, запропонована модель поєднує просторове групування користувацьких запитів, ідентифікацію мережевого трафіку та багатокритеріальний вибір Fog-вузла в єдиний адаптивний контур керування. Формалізовано вектор стану платформи, інтегральну функцію оцінювання вузлів та умову запуску адаптації. Розроблено дослідницький програмний продукт, який відтворює запропоновану архітектуру на локальному стенді типу central controller + fog agents. Пілотна апробація підтвердила працездатність підходу та можливість зниження середнього часу відповіді і кількості порушень SLA у сценаріях змішаного навантаження.

Ключові слова: мікросервісна архітектура, телекомунікаційні мережі, Edge computing, Fog computing, адаптивне керування, кластеризація користувачів, ідентифікація трафіку, міграція сервісів.

KORETSKYI Oleksandr
State University of Information and Communication Technology

ARCHITECTURAL MODEL OF THE METHOD FOR CONSTRUCTING A MICROSERVICE SOFTWARE PLATFORM IN THE EDGE/FOG ENVIRONMENT

The paper proposes an architectural model of a microservice software platform for telecommunication networks in an Edge/Fog environment. In contrast to static service placement approaches, the proposed model integrates spatial grouping of user requests, network traffic identification, and multi-criteria Fog node selection into a single adaptive control loop. The platform state vector, the integral node evaluation function, and the adaptation trigger condition are formalized. A research software prototype was developed to reproduce the proposed architecture on a local central controller + fog agents testbed. Pilot testing confirmed the feasibility of the approach and showed a potential reduction in average response time and SLA violations under mixed-load scenarios.

Keywords: microservice architecture, telecommunication networks, Edge computing, Fog computing, adaptive control, user clustering, traffic identification, service migration.

Стаття надійшла до редакції / Received 07.04.2026
Прийнята до друку / Accepted 30.04.2026
Опубліковано / Published 31.05.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Розвиток телекомунікаційних мереж нового покоління супроводжується зростанням кількості підключених пристроїв, неоднорідністю трафіку та підвищенням вимог до затримки оброблення сервісних запитів. У таких умовах централізоване або статичне розміщення мікросервісів не завжди забезпечує необхідну швидкодію, оскільки навантаження в мережі змінюється нерівномірно у просторі та часі.

Особливо складною є ситуація для Edge/Fog-середовищ, де обчислювальні ресурси розподілені між периферійними вузлами, а якість обслуговування залежить не лише від поточного завантаження вузла, а й від класу трафіку, просторового розташування користувачів, обмежень SLA та вартості міграції сервісу. Тому програмна платформа повинна не тільки запускати мікросервіси, а й безперервно аналізувати стан мережевого середовища та приймати рішення щодо масштабування, реконфігурації або міграції.

Актуальним науково-практичним завданням є побудова інтегрованої архітектурної моделі, у якій кластеризація користувацьких запитів, ідентифікація трафіку та вибір Fog-вузла не розглядаються як ізольовані модулі, а функціонують у межах єдиного адаптивного контуру керування мікросервісним програмним забезпеченням.

Аналіз останніх досліджень і публікацій

Питання розподіленого виконання сервісів на межі мережі активно досліджується у працях, присвячених Fog та Edge computing. У роботах [1]–[4] показано, що перенесення оброблення ближче до джерела даних є важливим для IoT-, 5G/6G- та інших сценаріїв, чутливих до затримки. Разом з тим такі середовища характеризуються гетерогенністю вузлів, обмеженістю ресурсів і потребою в адаптивному розміщенні сервісів.

Мікросервісна архітектура забезпечує декомпозицію прикладних функцій, незалежне розгортання

компонентів і гнучке масштабування [5]. Для телекомунікаційних мереж це створює основу для переміщення окремих сервісних функцій між Edge/Fog-вузлами без перебудови всієї системи. Однак сама мікросервісна декомпозиція не вирішує задачу вибору найдоцільнішого вузла виконання.

У дослідженнях сервісного розміщення в Fog-середовищах підкреслюється багатокритеріальний характер задачі: необхідно враховувати затримку, доступність, CPU, RAM, енергетичні обмеження, вартість міграції та профіль сервісу [9]–[11]. Для аналізу користувацького попиту застосовуються методи кластеризації, зокрема k-середніх [6], а для класифікації трафіку - рекурентні нейронні мережі, зокрема LSTM [8].

Невирішеною частиною проблеми залишається інтеграція цих механізмів у єдину архітектуру програмної платформи. Більшість відомих рішень розглядає кластеризацію користувачів, ідентифікацію трафіку та сервісне розміщення як окремі задачі. Запропонована в цій статті модель спрямована на їх узгодження в межах одного циклу: збір стану - оцінювання - вибір стратегії - оркестрація - зворотний зв'язок.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є розроблення архітектурної моделі програмної платформи мікросервісного програмного забезпечення для телекомунікаційних мереж в Edge/Fog-середовищі, яка інтегрує просторове групування користувацьких запитів, ідентифікацію мережевого трафіку та багатокритеріальний вибір Fog-вузла в єдиний адаптивний контур керування сервісними функціями.

Для досягнення мети необхідно: визначити логічну структуру платформи; формалізувати вектор стану та функцію оцінювання Fog-вузлів; описати механізм прийняття адаптивного рішення; реалізувати дослідницький програмний продукт і провести пілотну апробацію на локальному стенді.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Запропонована архітектурна модель орієнтована на підтримку життєвого циклу мікросервісів у динамічному Edge/Fog-середовищі. Її особливість полягає в тому, що рішення щодо розміщення або міграції сервісу приймається не за одним критерієм, а на основі поєднання просторового, трафікового та ресурсного контекстів.

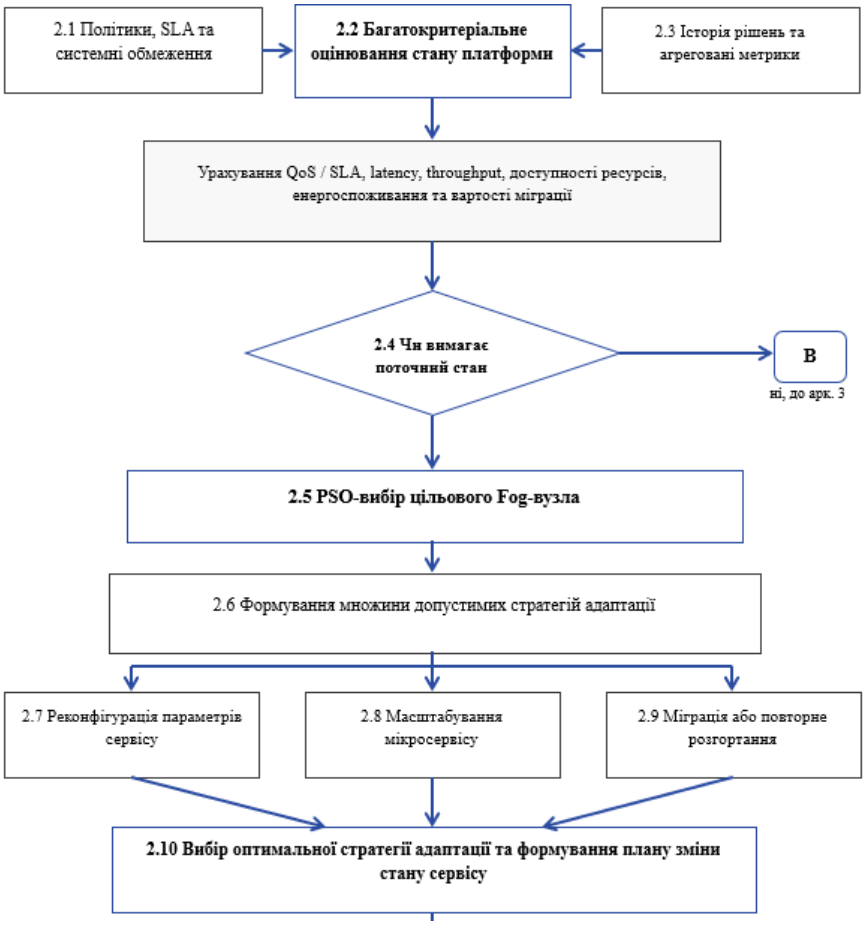

Рис. 1. Узагальнена архітектурна модель програмної платформи

Архітектура включає три взаємопов'язані рівні. Перший рівень відповідає за збір даних, попереднє оброблення подій і формування вектора стану платформи. Другий рівень реалізує багатокритеріальне

оцінювання, ранжування Fog-вузлів і вибір стратегії адаптації. Третій рівень забезпечує оркестрацію мікросервісів, виконання адаптивної дії та перевірку її фактичного ефекту через зворотний зв'язок.

Множина користувацьких запитів у поточному часовому вікні подається як:

$$R_t = \{ r_i = (x_i, y_i, q_i) \mid i = 1, 2, \dots, m_t \}, \quad (1)$$

де r_i - і-й запит користувача; x_i, y_i - просторові координати джерела запиту; q_i - тип або ідентифікатор сервісної функції; m_t - кількість активних запитів у момент часу t.

Для виявлення зон скупчення однотипних запитів використовується кластеризація за методом k-середніх. Її цільова функція має вигляд:

$$J = \Sigma_{j=1}^{k} \Sigma_{ri} \in C_j \,||\, p_i - \mu_j \,||^2 \rightarrow min, \quad (2)$$

де C_j - j-й кластер; $p_i = (x_i, y_i)$ - координати запиту; μ_j - центр відповідного кластера; k - кількість зон скупчення. Результатом цього етапу є просторовий контекст попиту, який використовується під час вибору цільового Fog-вузла.

Після групування запитів, ідентифікації трафіку та збору телеметрії формується інтегрований вектор стану платформи:

$$S(t) =< \tau_t, \pi_t, C_t, N_t, L_t, A_t >, \quad (3)$$

де τ_t - клас мережевого трафіку; π_t - пріоритет сервісного запиту; C_t - параметри активного користувацького кластера; N_t - множина доступних Fog-вузлів та їхніх метрик; L_t - обмеження SLA; A_t - множина допустимих адаптивних дій.

Багатокритеріальне оцінювання Fog-вузлів здійснюється за інтегральною функцією вартості:

$$Cost(n) = w^1 D_n + w^2 CPU_n + w^3 RAM_n + w^4 Geo_n + w^5 M_n, \quad (4)$$

де D_n - нормалізована затримка до вузла n; CPU_n та RAM_n - рівні завантаження ресурсів; Geo_n - просторовий чинник відносно активного кластера користувачів; M_n - очікувана вартість міграції; $w^1, \dots, w^5$ - вагові коефіцієнти, що можуть змінюватися залежно від класу трафіку та SLA-профілю.

Цільовий Fog-вузол визначається правилом мінімізації інтегральної вартості:

$$n* = argmin_n \in N_t Cost(n). \quad (5)$$

Таким чином, вибір вузла ґрунтується не лише на мінімальній затримці, а й на балансі ресурсів, просторовій близькості до активного кластера та доцільності міграції.

Рішення про запуск адаптації приймається за наявності хоча б однієї з умов:

$$Adapt(t) = 1, \text{якщо} D_t > D_{max} \text{або} U_n > U_{max} \text{або} SLA_t = 0; \text{інакше} Adapt(t) = 0. \quad (6)$$

Тут D_t - фактична затримка обслуговування; D_{max} - допустима затримка; U_n - рівень завантаження вузла; U_{max} - граничне навантаження; SLA_t - індикатор виконання SLA. Якщо $Adapt(t) = 1$, платформа формує одну з дій: збереження поточного стану, реконфігурація сервісу, масштабування, міграція або повторне розгортання.

Узагальнені модулі запропонованої архітектури подано в табл. 1.

Таблиця 1

Основні модулі архітектурної моделі платформи

Модуль	Призначення	Результат
User Request Grouping	Просторове групування однотипних користувацьких запитів	кластери запитів, центри скупчення
Traffic Identification	Визначення класу трафіку та профілю обслуговування	traffic class, priority, SLA profile
Fog Node Selection	Багатокритеріальне ранжування кандидатних Fog-вузлів	selected node, score
Decision Core	Формування адаптивного рішення на основі S(t)	adaptation strategy, execution plan
Orchestrator	Виконання дії: запуск, масштабування або міграція сервісу	placement update, migration event
Feedback Loop	Перевірка досягнення цільових показників після адаптації	stable state або repeated cycle

Наукова новизна запропонованої архітектури полягає у тому, що часткові механізми аналізу попиту, трафіку та ресурсного стану об'єднуються не послідовно і формально, а в єдиний замкнений контур. Просторове групування визначає, де формується концентрація попиту; ідентифікація трафіку визначає, який профіль обслуговування потрібний; багатокритеріальний вибір Fog-вузла визначає, де доцільно виконувати сервіс; feedback loop перевіряє, чи справді прийняте рішення покращило стан системи.

Програмна реалізація та апробація

Для перевірки запропонованої архітектурної моделі розроблено дослідницький програмний продукт adaptive_fog_platform. Реалізація виконана на основі Python, FastAPI, SQLite, SQLAlchemy, NumPy, scikit-learn та psutil. Програмний стенд побудовано за схемою central controller + fog agents: центральний контролер збирає метрики, формує вектор стану та приймає рішення, а Fog-агенти імітують периферійні вузли, виконують тестові сервіси та передають телеметрію.

У межах реалізації виділено три логічні групи компонентів: layer1_state_collection для збору стану та кластеризації; layer2_decision для оцінювання, ранжування та планування адаптації; layer3_orchestration для виконання рішень, моніторингу QoS та збереження результатів. Такий поділ відповідає запропонованій

трирівневій архітектурі та спрощує подальше масштабування платформи.

Експериментальна апробація проводилася у двох режимах. Режим compare використовувався для прямого порівняння базового підходу без новизни та запропонованого режиму proposed. Режим stage-compare дозволяв поетапно вмикати окремі компоненти новизни: групування користувачів, вибір Fog-вузла та координацію на основі класу трафіку.

Оцінювання виконувалося за метриками, наведеними в табл. 2.

Таблиця 2

Метрики експериментальної оцінки

Метрика	Позначення	Інтерпретація
Середній час відповіді	avg_response_time_ms	менше значення є кращим
95-й перцентиль затримки	p95_response_time_ms	менше значення є кращим
Пропускна здатність	throughput_rps	більше значення є кращим
Кількість порушень SLA	sla_violations	менше значення є кращим
Кількість міграцій	migration_count	менше значення є кращим за однакового QoS
Індекси балансування CPU/RAM	cpu_balance_index, ram_balance_index	значення, ближче до 1, свідчить про кращий баланс

Для кількісного порівняння режимів використовувався показник відносного поліпшення цільової метрики:

$$I = \big((Mbase - Mprop)/Mbase\big) \cdot 100$$

де $Mbase$ - значення метрики в базовому режимі; $Mprop$ - значення тієї самої метрики для запропонованої платформи. Для метрик типу latency, p95 та SLA violations додатне значення I відповідає покращенню. Для throughput інтерпретація виконується з урахуванням того, що більше значення є кращим.

У пілотному сценарії змішаного навантаження mixed_city запропонований режим продемонстрував скорочення середнього часу відповіді приблизно на 5-6 % та зменшення кількості порушень SLA на 3 події порівняно з базовим режимом without_novelty. Отриманий результат не слід розглядати як остаточну статистичну оцінку для промислового середовища, оскільки апробація виконувалася на локальному стенді. Водночас він підтверджує працездатність архітектури та доцільність інтеграції трьох компонентів у єдиний адаптивний контур.

Режим stage-compare показав, що окремі модулі новизни мають різний характер впливу. Кластеризація користувачів насамперед покращує просторову структурованість попиту; багатокритеріальний вибір Fog-вузла впливає на обґрунтованість розміщення сервісу; ідентифікація трафіку коригує SLA-профіль і вагові коефіцієнти функції Cost(n). Найбільш цілісний ефект досягається у режимі proposed, де всі компоненти працюють спільно.

Водночас результати залишаються чутливими до кількості Fog-вузлів, параметрів навантаження, вагових коефіцієнтів, порогів міграції та топологічних особливостей стенда. Це визначає напрями подальшого розвитку: розширення набору сценаріїв, збільшення кількості вузлів, калібрування evaluator та перевірка моделі у фізично розподіленому середовищі.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

1. У статті запропоновано архітектурну модель програмної платформи методу побудови програмної платформи мікросервісного програмного забезпечення в Edge/Fog -середовищі. Модель інтегрує просторове групування користувацьких запитів, ідентифікацію мережевого трафіку та багатокритеріальний вибір Fog-вузла в єдиний адаптивний контур керування.

2. Сформовано трирівневу логіку платформи: рівень збору стану та побудови $S(t)$, рівень багатокритеріального прийняття рішень і рівень оркестрації зі зворотним зв'язком. Такий підхід дозволяє перейти від статичного розміщення мікросервісів до контекстно-залежного керування їх виконанням.

3. Формалізовано основні елементи моделі: множину користувацьких запитів, критерій просторової кластеризації, вектор стану платформи, інтегральну функцію вартості Fog-вузла та умову запуску адаптації.

4. Розроблено дослідницький програмний продукт, який відтворює запропоновану модель на локальному стенді central controller + fog agents і підтримує порівняння базового та запропонованого режимів керування.

5. Пілотна апробація підтвердила працездатність підходу та показала можливість сценарного покращення якості обслуговування у вигляді зменшення середнього часу відповіді та кількості порушень SLA. Практичне значення результатів полягає у можливості використання платформи як дослідницького стенда для подальшої верифікації методів розміщення, міграції та координації мікросервісів у телекомунікаційних мережах.

References

1. Bonomi F., Milito R., Zhu J., Addepalli S. Fog Computing and Its Role in the Internet of Things. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. Helsinki, 2012. P. 13-16. DOI: 10.1145/2342509.2342513.

2. Chiang M., Zhang T. Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal. 2016. Vol. 3, No. 6. P. 854-864. DOI: 10.1109/JIOT.2016.2584538.

3. Satyanarayanan M. The Emergence of Edge Computing. Computer. 2017. Vol. 50, No. 1. P. 30-39. DOI: 10.1109/MC.2017.9.

4. Varghese B., Wang N., Barbhuiya S., Kilpatrick P., Nikolopoulos D. S. Challenges and Opportunities in Edge Computing. Proceedings of the 2016 IEEE International Conference on Smart Cloud. New York, 2016. P. 20-26. DOI: 10.1109/SmartCloud.2016.18.

5. Dragoni N., Giallorenzo S., Lafuente A. L., Mazzara M., Montesi F., Mustafin R., Safina L. Microservices: Yesterday, Today, and Tomorrow. Present and Ulterior Software Engineering. Cham: Springer, 2017. P. 195-216. DOI: 10.1007/978-3-319-67425-4_12.

6. Lloyd S. Least Squares Quantization in PCM. IEEE Transactions on Information Theory. 1982. Vol. 28, No. 2. P. 129-137. DOI: 10.1109/TIT.1982.1056489.

7. Kennedy J., Eberhart R. Particle Swarm Optimization. Proceedings of ICNN'95 - International Conference on Neural Networks. Perth, 1995. Vol. 4. P. 1942-1948. DOI: 10.1109/ICNN.1995.488968.

8. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997. Vol. 9, No. 8. P. 1735-1780. DOI: 10.1162/neco.1997.9.8.1735.

9. Sadatacharapandi T. P., Padmavathi S. Survey on Service Placement, Provisioning, and Composition for Fog-Based IoT Systems. International Journal of Cloud Applications and Computing. 2022. Vol. 12, No. 1. DOI: 10.4018/IJCAC.305212.

10. Abid M., Hnida C., Kessentini M., Kacem A. H., Hadj Kacem A. MicroFog: A Framework for Scalable Placement of Microservices-Based IoT Applications in Federated Fog Environments. Journal of Systems and Software. 2024. Vol. 208. Art. 111910. DOI: 10.1016/j.jss.2023.111910.

11. Gasmi K., Dilek S., Tosun S., Ozdemir S. A Survey on Computation Offloading and Service Placement in Fog Computing-Based IoT. The Journal of Supercomputing. 2022. Vol. 78, No. 4. P. 4705-4739. DOI: 10.1007/s11227-021-03941-y.