УДК: 004.724.4(045)

СПОСІБ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В SDN МЕРЕЖАХ НА ОСНОВІ ACS З УДОСКОНАЛЕНОЮ ДИНАМІЧНОЮ ВІДМОВОСТІЙКІСТЮ

аспірант, Щур В. Ю.

https://orcid.org/0000-0001-8925-4813

email: vadimmshchur@gmail.com

доктор технічних наук, професор, Кулаков Ю. О.

https://orcid.org/0000-0002-8981-5649

email: ya.kulakov@gmail.com

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна, Київ.

В статті розглянуто актуальне питання балансування навантаження в розподілених обчислювальних системах. Проведено аналіз існуючих рішень, визначено завдання, проблематику і практичне значення. Запропоновано удосконалений спосіб балансування з використанням методу контрольних точок та додаткового фактору довіри, який дозволив гарантувати рівномірне навантаження на контролери, зберігаючи прийнятний рівень ефективності. Проведено оцінку продуктивності та аналітичне порівняння запропонованого методу з існуючими алгоритмами, а також окреслено кроки для подальших досліджень.

Ключові слова: балансування навантаження, обчислювальні системи, оцінка продуктивності, алгоритм, динамічна відмовостійкість, ефективність.

аспирант Щур В. Ю., доктор технических наук, профессор Кулаков Ю. А. Способ балансировки нагрузки в SDN сетях на основе

ACS с усовершенствованной динамической отказоустойчивостью / Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Украина, Киев.

В статье рассмотрен актуальный вопрос балансировки нагрузки в распределенных вычислительных системах. Проведен анализ существующих решений, определены задачи, проблематика и практическое значение. Предложен усовершенствованный способ балансировки с использованием метода контрольных точек и дополнительного фактора доверия, который позволил обеспечить равномерную нагрузку на контроллеры, сохраняя приемлемый уровень эффективности. Проведена оценка производительности и аналитическое сравнение предложенного метода с существующими методами, а также обозначены шаги для дальнейших исследований.

Ключевые слова: балансировка нагрузки, вычислительные системы, оценка производительности, алгоритм, динамическая отказоустойчивость, эффективность.

V. Y. Shchur, Y. A. Kulakov, ScD of Computer Science. Load balancing method in SDN networks based on ACS with improved dynamic resiliency / National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, Kyiv.

The article discusses the topical issue of load balancing in distributed computing systems. The analysis of existing solutions is carried out, tasks, problems and practical significance are determined. An improved balancing method using the checkpoint method and an additional confidence factor is proposed, which made it possible to ensure a uniform load on the controllers, while maintaining an acceptable level of efficiency.

An assessment of the performance and comparison of the proposed method with existing methods is carried out, as well as steps for further research are indicated.

Keywords: load balancing, computing systems, performance evaluation, algorithm, dynamic fault tolerance, efficiency.

Вступ. Ідея технології SDN не є новою, проте активне впровадження технології припадає лише на останні 10 років. Однією із важливих задач при організації мережі є задача балансування трафіку, що включає в собі маршрутизацію та контроль трафіку під час передач (забезпечення рівномірного завантаження каналів). Використання нової технології вимагає розробки нових алгоритмів та протоколів, або адаптації традиційних, тому задача балансування навантаження є дуже актуальною.

У той же час, існує дуже мало методів балансування навантаження, які дозволяють гарантувати рівномірне навантаження на контролери в мережі SDN, зберігаючи прийнятний рівень ефективності.

Аналіз останніх досліджень і публікацій. Протягом останніх років з’являється все більше статей присвячених темі розподілу навантаження, зокрема, залежні від часу завершення завдання на машині розробляли і вдосконалили такі вчені як Keshav, Elzeki, Ramadhika Dewanto, Rendy Munadi, алгоритми грубої сили і засновані на статистичних даних розглядалися в роботах таких вчених як Hu, Ghanbari, Chen, Liu, Yao-Chun Wang, алгоритми, засновані на біологічних феномени розглядалися Mishra, Dhinesh, Jamil S. Al Azzeh, Abdelwadood Mesleh, а також багато інших дослідників, що працюють над проблемами розподілу навантаження. Проте

запропоновані методи, не показують достатньої ефективності, щоб задовольнити поставлені задачі.

Постановка завдання. Завданням є зменшення затримки при ремаршрутизації та забезпечення більш рівномірного завантаження мережі шляхом балансування навантаження, з використанням можливостей програмо-конфігурованих мереж.

Практичне значення. Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- вдосконалення методів маршрутизації;
- аналіз та прогнозування трафіку в SDN;
- балансування навантаження в SDN мережах.

Виклад основного матеріалу. У даній статті пропонується *удосконалена динамічна відмовостійкість* на основі ACS (*EDAFT*) - розширена версія алгоритму, запропонованого [1], яка заснована на поведінці мурашиної колонії в пошуках джерела їжі шляхом побудови оптимального шляху між гніздом і джерелом їжі. Ця аналогія аналогічна процесу побудови оптимального шляху між завданнями і ресурсами в сітковій обчислювальній системі. У пропонованому алгоритмі цей процес додатково розширено для того, щоб мурахи могли виконувати дослідження ресурсів в процесі повторної відправки на основі контрольних точок, щоб призначити будь-яке невиконане завдання альтернативним ресурсам з більш високою ймовірністю успіху. Щоб додатково поліпшити техніку поновлення феромону, вводиться фактор довіри, щоб винагороджувати придатні ресурси або штрафувати непридатні ресурси з урахуванням історії виконання, щоб управляти зменшенням або збільшенням феромону. Очікується, що поліпшена формула поновлення феромону буде належним чином управляти призначенням завдань на основі придатності ресурсів, що в кінцевому підсумку може знизити

ймовірність збою. Під час початкової відправки завдання кожен ресурс повинен мати заздалегідь визначені параметри, такі як швидкість процесора, поточне навантаження, а також пропускну здатність і кількість елементів обробки. Всі ці параметри будуть використовуватися для розрахунку початкового значення феромона (PVrj) для кожної комбінації ресурсу r і завдання j. Початкова формула значення феромона дається наступним рівнянням (1.1):

$$PV_{rj} = \left[\frac{S_j}{\text{bandwidthr}} + \frac{C_j}{\text{MIPSr}(1-\text{loadr})}\right]^{-1} \qquad (1.1)$$

Де S_j - це розмір, а C_j - необхідна обчислювальна потужність для даної задачі j, $bandwidth_j$ - це доступна пропускна здатність ресурсу r, $MIPS_r$ - швидкість процесора, а $load_r$ - поточна навантаження на ресурс r. Також потрібно звернути увагу, що початкове значення феромона призначається під час ініціалізації, але після цього воно розглядається як значення феромона ресурсу. Оскільки початкове значення феромона розраховується для кожної комбінації ресурсу і завдання, ця інформація зберігається в PV_{matrix}, як в (1.2):

$$PV_{matrix} = \begin{bmatrix} PV_{1,1} & PV_{1,2} & PV_{1,n-1} & PV_{1,n} \\ PV_{2,1} & PV_{2,2} & PV_{2,n-1} & PV_{2,n} \\ PV_{m-1,1} & PV_{m-1,2} & PV_{m-1,n-1} & PV_{m-1,n} \\ PV_{m,1} & PV_{m,2} & PV_{m,n-1} & PV_{m,n} \end{bmatrix} \qquad (1.2)$$

де n - загальна кількість завдань, а m - загальна кількість ресурсів. PV_{matrix} - це логічна форма топології мурахи, при якій мураха переміщається з одного індексу в інший, щоб знайти кращий ресурс для призначення завдань. Передбачається, що всі ресурси взаємопов'язані, що означає, що якщо завдання відбувається з певного ресурсу, вона може бути призначена всім іншим доступним ресурсам. Кожен рядок в PV_{matrix} представляє список можливих завдань для ресурсу r, в той час як кожен стовпець представляє список можливих ресурсів для завдання j.

Найбільше значення феромону у кожному стовпці мурахи вважатимуть найбільш підходящим ресурсом, і завдання буде передано ресурсу з найвищим феромоном для обробки. Як тільки завдання буде призначено, значення феромону у PV_{matrix} буде оновлено глобальним оновленням феромонів (1.3), щоб зменшити кількість феромонів, присвоєних поточному ресурсу, щоб воно стало менш привабливим до наступної мурахи і призвело до розвідки інших ресурсів. tr_j - кількість феромонів на ресурсі, тоді як Δtr_j - 1 / L_{best}, де L_{best} позначає тривалість найкращої глобальної екскурсії або іншим чином (кращий глобальний тур не знайдено), $\Delta tr_j = 0$.

$$\tau_{rj} = (1 - \rho) \cdot \tau_{rj} + \rho \cdot \Delta\tau_{rj} \qquad (1.3)$$

ρ - швидкість випаровування, яке динамічно контролюється за допомогою наступної формули (1.4) з m і n в якості загальної кількості ресурсів і завдань відповідно:

$$\rho = \left[\left(\frac{n}{m}\right)^2\right]^{-1} \qquad (1.4)$$

Типовий алгоритм ACS складається з глобальних і локальних оновлень феромонів. У EDAFT покращено локальне оновлення феромону, щоб включити фактор довіри, так що додається більше феромону, якщо ресурс завершить виконання завдання або іншим чином випарується існуючий феромон. Поліпшене глобальне оновлення феромону (1.5) має наступний вигляд:

$$\tau_{rj} = (1 - \rho) \cdot \tau_{rj} + [\rho \cdot \tau 0\ (R_H)]^C \qquad (1.5)$$

де ρ - швидкість випаровування, trj - поточна інтенсивність феромона для ресурсу r, т0 - початкове значення феромона ресурсу r, C - коефіцієнт довіри, який визначається або завершенням завдання (C = 1.5), або невдачею завдання (C = 1.0), Певне значення C в цьому експерименті забезпечує найменше стандартне відхилення

балансування навантаження. Коефіцієнт довіри може варіюватися в залежності від рівня чутливості довіри, який повинен бути застосований, при якому занадто велике покарання може призвести до того, що непридатні ресурси ніколи не будуть призначені для задач після збою, або занадто багато стимулів може привести до того, що відповідні ресурси будуть призначені для більшості завдань. R_H - це середньозважена історія виконання ресурсу r і розраховується за рівнянням (1.6):

$$R_H(i) = \begin{cases} R_T(i) = \frac{CP_{success}}{CP_{failed}+CP_{success}}, i = 0 \\ (1-\alpha)\cdot R_T(i) + \alpha \cdot R_H(i-1), i > 0 \end{cases} \quad (1.6)$$

де R_T (i) - поточна історія виконання при дублі i, $CP_{success}$ вказує поточний успішний виклик контрольної точки, а CP_{failed} - поточна невдала контрольна точка в ресурсі r відповідно. Для кожного ресурсу r і спочатку встановлений на 0 і буде збільшуватися на 1 для кожного локального процесу оновлення феромону, R_H (i-1) - це раніше записана історія виконання, а α - ступінь зменшення ваги, встановлена на 0,5. Історія виконання (також відома як придатність ресурсів) буде використовуватися для контролю кількості феромонів, які будуть випаруватьсяабо посилені на відповідному ресурсі, і в кінцевому підсумку допоможе наступним мурашкам визначити кращі ресурси під час призначення завдання; чим краще історія виконання, тим вище кількість призначених завдань. На рисунку 1 показаний робочий процес EDAFT високого рівня, запропонований в [1, 2] з поліпшеним процесом, виділеним для ясності.

Фаза 1

1) ***(Початок)*** Розрахувати початкове значення феромону для визначення стану всіх ресурсів;

2) Вибір найкращого ресурсу з найбільшим феромоном;

3) Застосувати глобальне оновлення феромонів;

4) Виконати завдання ресурсом з найбільшим значенням феромомону.

Фаза 2

1) Якщо завдання *виконано* – локальне збільшення феромону зі стимулом для ресурса, *інакше* – оновлення феромону зі штрафом до ресурса (перейти до Фази 3). У будь якому випадку – звільнення ресурсу;

2) Якщо *виконані* всі завдання – ***(Кінець)***, *інакше* – локальне оновлення феромонів (перейти до Фази 1).

Фаза 3

1) Якщо завдання було *провалено*:

1.1) отримати інформацію про контрольну точку;

1.2) збільшити кількість відмов ресурсу;

1.3) повторити завдання з останнього збереженого стану;

1.4) локальне оновлення феромону (перейти до Фази 1).

інакше:

1.5) зберегти інформацію про контрольну точку;

1.6) збільшення кількості успіху ресурсу;

1.7) локальне оновлення феромону (перейти до Фази 1, п.4.)

Для кожного завдання в черзі буде згенеровано мураха для пошуку ресурсів на основі значень феромону. Перед відправкою першого завдання в черзі буде розраховане початкове значення феромона для визначення стану всіх ресурсів. Вибір ресурсу буде виконуватися на основі рівнів феромону, або з початкового розрахунку феромону, або з процесу оновлення феромону. Як тільки завдання призначено будь-якому ресурсі, мураха застосує глобальне оновлення феромону, щоб зменшити кількість феромону, щоб ресурс став менш привабливим для наступної мурахи. Кожне призначене завдання буде розділено на кілька тимчасових контрольних точок,

записаних під час виконання. У разі невдачі завдання буде піддано процесу перепланування і буде призначено альтернативний ресурс з останньої збереженої контрольної точки і локальне оновлення феромону зі штрафом буде застосовано до ресурсу, який не зміг зменшити інтенсивність феромона.

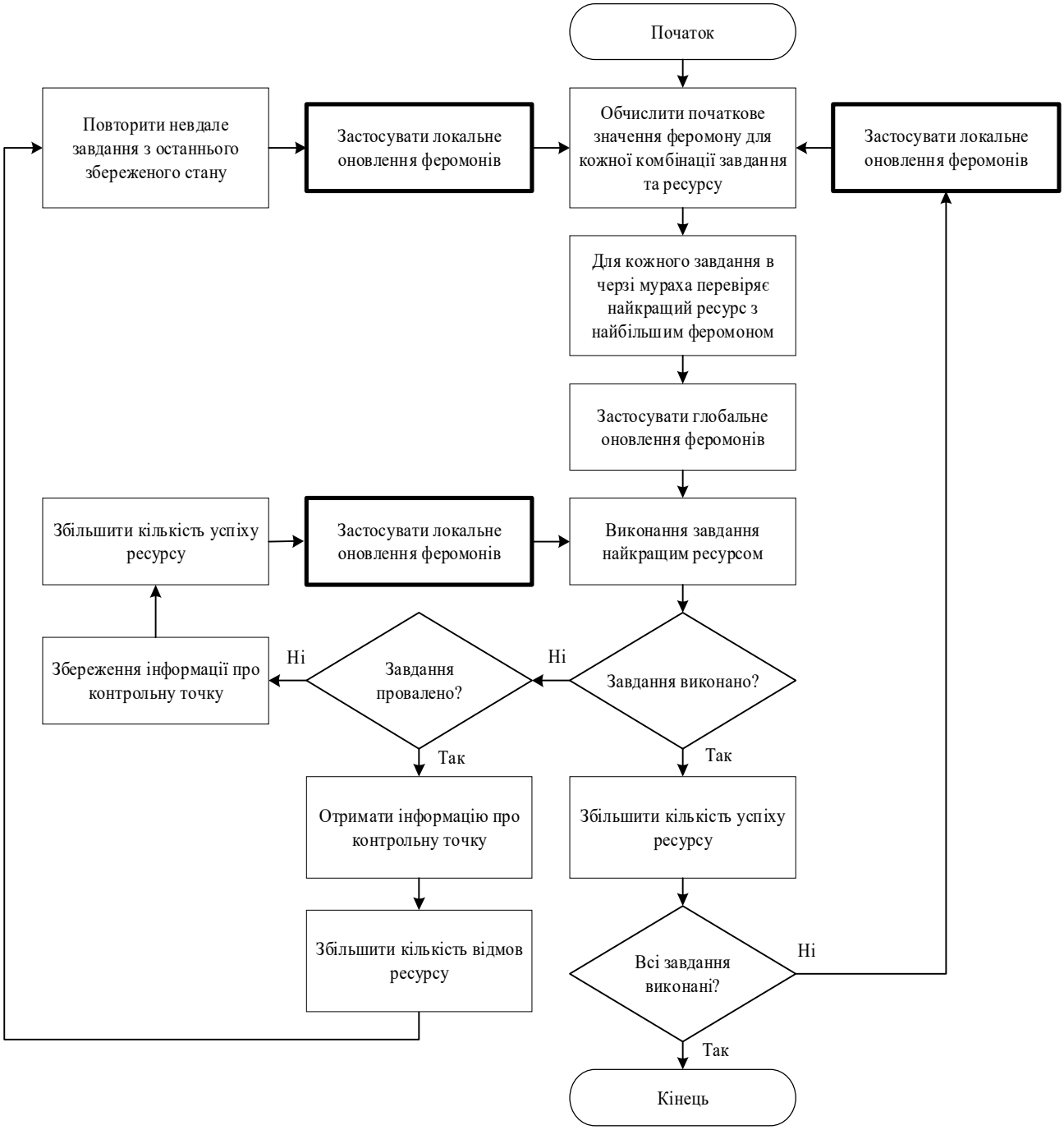


Рис. 1. Модифікований алгоритм EDAF

Якщо виконання виявилась успішною, локальне оновлення феромону з стимулом буде застосовано до ресурсу для збільшення феромона. Як у випадку збою виконання, так і в разі успіху виконання ресурс буде звільнений для наступного призначення завдання після процесу оновлення феромону.

Результати. Запропонований алгоритм порівнювався з TACO [3], FTACO [4], ACOwFT і ACO [5], де всі алгоритми були перероблені на основі опублікованих псевдокодів, формулювань і блок-схем. Вони придатні для використання при перевірці, тому що алгоритм EDAFT натхненний усіма алгоритмами з точки зору методів відмовостійкості. Всі експерименти моделюються в імітаторі на основі JAVA, відомому як GridSim Toolkit, тому що він забезпечує комплексне оточення імітованої сітки, в яку включені більшість компонентів. Кожен алгоритм виконується 10 раз для кожного діапазону несправностей, і середнє значення береться для більш точного вимірювання.

Список показників продуктивності включає час виконання, середній час виконання і затримку на завдання, балансування навантаження і коефіцієнт успішності виконання. *Час виконання* для всіх алгоритмів майже однаковий, коли немає помилок (див. рис. 2). Однак у міру збільшення частоти відмов час виконання для EDAFT є найнижчим серед всіх і супроводжується ACOwFT і FTACO відповідно.

Вкрай важливо зберегти час виконання, незважаючи на наявність збою, щоб гарантувати, що пропускна здатність також може бути збережена. Як показано на рисунку 3, *середній час виконання* для EDAFT, FTACO і ACOwFT практично однаковий в порівнянні з TACO і ACO. Саме тут техніка контрольних точок грає роль, оскільки вона дозволяє виконувати кожну невиконану завдання з останнього збереженого стану, а не з самого початку, що в кінцевому підсумку

скорочує час виконання індивідуального завдання. Результати також показують, що EDAFT, FTACO і ACOwFT можуть контролювати час виконання, використовуючи техніку контрольних точок при наявності збоїв, так що кожна окрема задача може бути повністю виконана своєчасно.

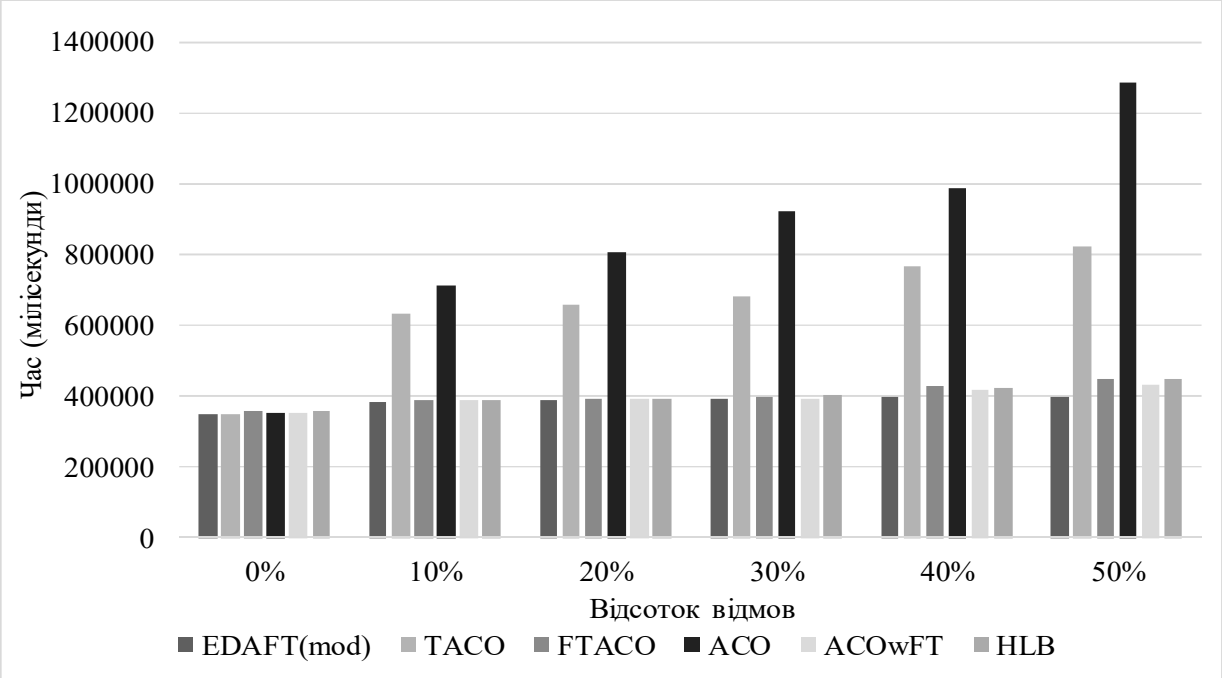
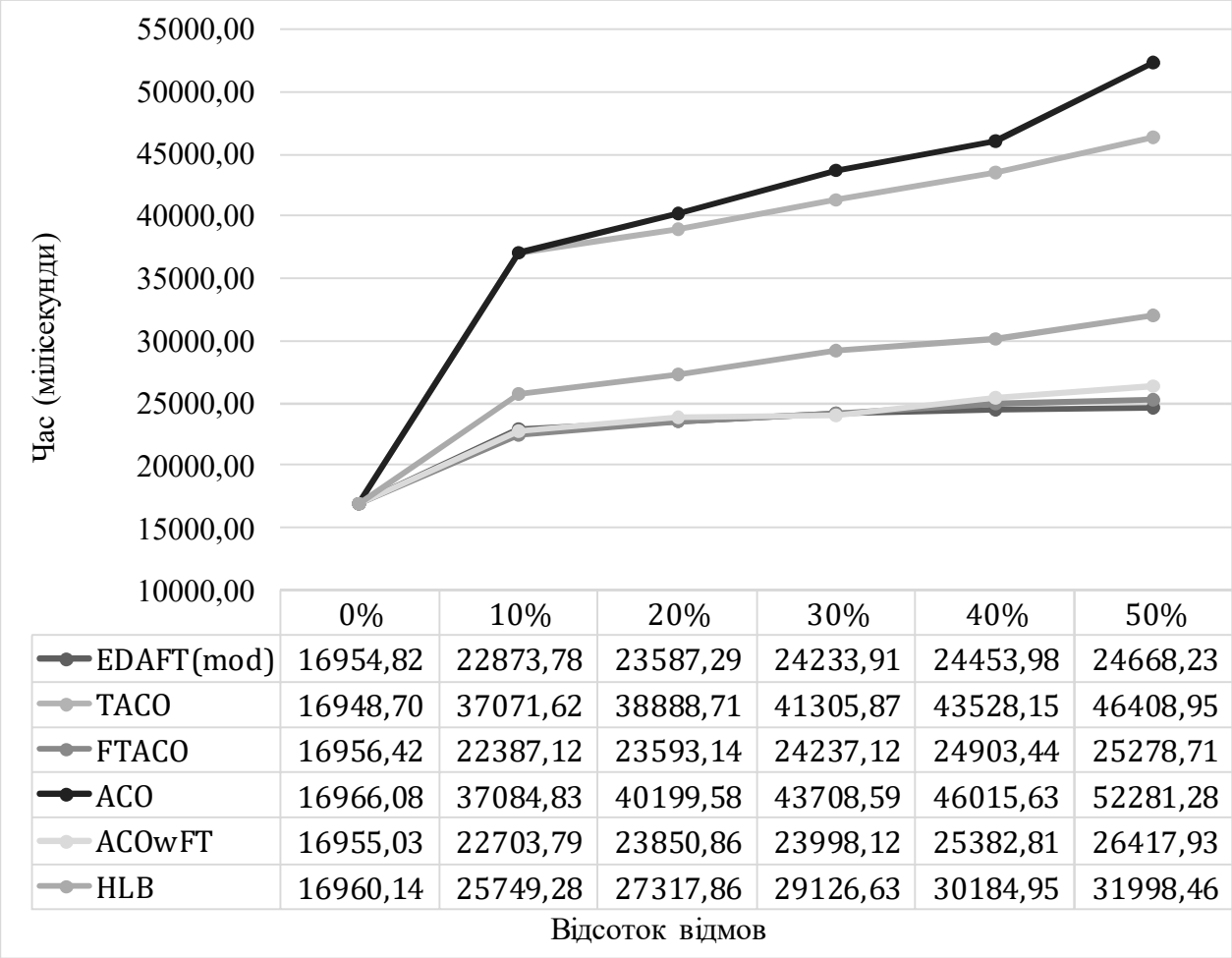


Рис. 2. Результати часу виконання

У будь-який відмовостійкій системі кінцевою метою є підтримка рівня *успішності виконання* без шкоди для продуктивності. На малюнку 4 показано, що EDAFT має більш високий рівень успіху в порівнянні з іншими алгоритмами.

Висновки. Внаслідок проведеного порівняння було виявлено, що розроблений алгоритм EDAFT має кращу загальну продуктивність в порівнянні з іншими алгоритмами з точки зору часу виконання (в середньому на 35%), пропускної здатності (на 38%), середнього часу виконання (на 26%), середньої затримки і швидкості успішного виконання. Проте, з точки зору розподілу навантаження, ACOwFT

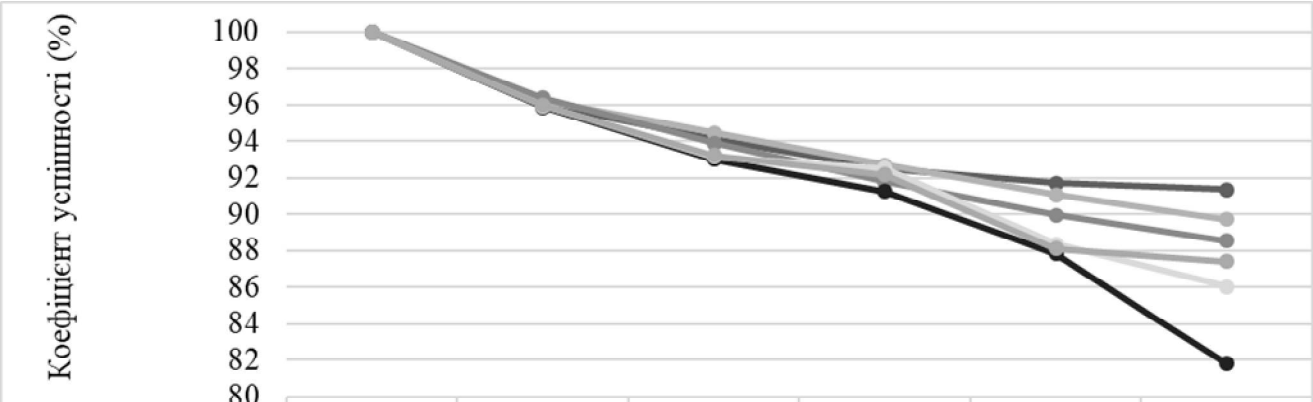
має кращу продуктивність з невеликою різницею в порівнянні з ACO та EDAFT.



	0%	10%	20%	30%	40%	50%
EDAFT(mod)	16954,82	22873,78	23587,29	24233,91	24453,98	24668,23
TACO	16948,70	37071,62	38888,71	41305,87	43528,15	46408,95
FTACO	16956,42	22387,12	23593,14	24237,12	24903,44	25278,71
ACO	16966,08	37084,83	40199,58	43708,59	46015,63	52281,28
ACOwFT	16955,03	22703,79	23850,86	23998,12	25382,81	26417,93
HLB	16960,14	25749,28	27317,86	29126,63	30184,95	31998,46

Рис 3. Результати середнього часу виконання завдання

Майбутні дослідження. Незважаючи на те, що EDAFT має в цілому гарну продуктивність, його можна ще поліпшити, додавши тимчасове припинення, щоб недавно збійному ресурсу не було призначено завдання, поки він не буде відновлений після збою. Ця можливість може бути ефективною, особливо коли розмір окремого завдання великий в певний момент часу, коли більшість ресурсів зайняті обробкою великого завдання, і тільки поточне призупинене завдання знаходиться в режимі очікування.



	0%	10%	20%	30%	40%	50%
EDAFT(mod)	100	96.05	94.12	92.52	91.71	91.38
TACO	100	96.23	94.45	92.71	91.09	89.71
FTACO	100	96.38	93.9	91.8	89.98	88.52
ACO	100	95.87	93.03	91.28	87.84	81.79
ACOwFT	100	95.93	93.15	92.55	88.37	86.04
HLB	100	95.97	93.24	92.17	88.11	87.42

Рис. 4. Результати успішності

Література:

1. Ramadhika Dewanto, Rendy Munadi, Ridha Muldina Negara. Improved Load Balancing on Software Defined Networkbased in Data Center Network <https://www.researchgate.net/publication/328777585_Improved_Load_Balancing_on_Software_Defined_Network-based_Equal_Cost_Multipath_Routing_in_Data_ Center_Network> (2018).

2. Jiafu Wan, Zhaogang Shu. Traffic Engineering in Software-Defined Networking. <https://ieeexplore.ieee.org/abstract/document/7496952> (2017).

3. Dania Marabissi, Romano Fantacci and Linda Simoncini. SDN-Based Routing for Backhauling in Ultra-Dense Networks

<https://www.researchgate.net/publication/ 332613701_SDN-Based_Routing_for_Backhauling_in_Ultra-Dense_Networks>(2019).

4. Yao-Chun Wang, Ying-Dar Lin, Guey-Yun Chang. SDN-based dynamic multipath forvarding for inter-data center networking <https://onlinelibrary.wiley.com/doi/full/ 10.1002/dac.3843> (2018).

5. A. Mayoral, R. Vilalta, R. Casellas, R. Muñoz, R. Martínez. Traffic Engineering enforcement in multi-domain SDN orchestration of Multi-Layer (packet/optical) networks <https://ieeexplore.ieee.org/abstract/document/7341810> (2018).

References:

1. Ramadhika Dewanto, Rendy Munadi, Ridha Muldina Negara. Improved Load Balancing on Software Defined Networkbased in Data Center Network <https://www.researchgate.net/publication/328777585_Improved_Load_Balancing_on_Software_Defined_Network-based_Equal_Cost_Multipath_Routing_in_Data_ Center_Network> (2018).

2. Jiafu Wan, Zhaogang Shu. Traffic Engineering in Software-Defined Networking: <https://ieeexplore.ieee.org/abstract/document/7496952> (2017).

3. Dania Marabissi, Romano Fantacci and Linda Simoncini. SDN-Based Routing for Backhauling in Ultra-Dense Networks <https://www.researchgate.net/publication/ 332613701_SDN-Based_Routing_for_Backhauling_in_Ultra-Dense_Networks>(2019).

4. Yao-Chun Wang, Ying-Dar Lin, Guey-Yun Chang. SDN-based dynamic multipath forvarding for inter-data center networking <https://onlinelibrary.wiley.com/doi/full/ 10.1002/dac.3843> (2018).

5. A. Mayoral, R. Vilalta, R. Casellas, R. Muñoz, R. Martínez. Traffic Engineering enforcement in multi-domain SDN orchestration of Multi-

Layer (packet/optical) networks
<https://ieeexplore.ieee.org/abstract/document/7341810> (2018).