https://doi.org/10.31891/2219-9365-2026-87-46
УДК 004.451:519.872

КОЗЕЛЬСЬКИЙ Олександр
Хмельницький національний університет
https://orcid.org/0009-0002-7157-6499
e-mail: oleksandr.kozelskiy@khmnu.edu.ua

# МЕТОД ЛОКАЛЬНОГО ПРОГНОЗУВАННЯ КАСКАДНИХ ПЕРЕВАНТАЖЕНЬ ЧЕРГ В ОПЕРАЦІЙНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

*У статті розглянуто задачу раннього виявлення каскадного накопичення повідомлень у послідовностях задач і черг операційних систем реального часу. На відміну від спостереження за окремою чергою, запропонований метод формує локальний підграф після першої небезпечної черги, оцінює швидкості надходження та обробки повідомлень, прогнозує поширення накопичення вздовж ланцюга та враховує можливу зміну режимів роботи системи. У результаті визначається не лише очікувана залишкова затримка, а й імовірність завершення обробки до встановленого граничного строку.*

*Метод перевірено на імітаційних сценаріях із різною кількістю взаємопов'язаних задач. Порівняння з простим пороговим контролем заповнення черги показало, що запропонований підхід значно краще виявляє майбутні прострочення та забезпечує більший запас часу для реагування. Дворівнева схема попередження дає змогу окремо формувати ранній ризиковий сигнал і критичне повідомлення з кращим співвідношенням між пропущеними подіями та хибними спрацьовуваннями.*

*Обмеженням дослідження є імітаційний характер перевірки та необхідність задавати максимально допустимий вік повідомлення. Водночас використані вхідні ознаки й критерії не залежать від служб конкретного ядра, що дає змогу застосовувати метод у різних операційних системах реального часу.*

*Ключові слова: операційна система реального часу, черга повідомлень, каскадне перевантаження, графи, експоненційне згладжування, метод Монте-Карло, граничний строк.*

KOZELSKYI Oleksandr
Khmelnytskyi National University

# METHOD FOR LOCAL PREDICTION OF CASCADING QUEUE OVERLOADS IN REAL-TIME OPERATING SYSTEMS

*The article addresses the problem of early detection of cascading message accumulation in sequences of tasks and queues in real-time operating systems. Unlike monitoring an individual queue, the proposed method constructs a local subgraph starting from the first hazardous queue, estimates message arrival and processing rates, predicts backlog propagation along the task–queue chain, and accounts for possible changes in the system operating mode. As a result, the method determines not only the expected residual processing delay, but also the probability that message handling will be completed before the specified deadline.*

*The method was evaluated using simulation scenarios with different numbers of interconnected tasks. Comparison with simple queue-fill threshold monitoring showed that the proposed approach detects future deadline violations more accurately and provides a larger time margin for corrective action. A two-level warning scheme makes it possible to generate an early risk signal separately from a critical alert, thereby achieving a better balance between missed hazardous events and false alarms. It also supports responses ranging from preventive load redistribution to immediate intervention when the predicted risk becomes unacceptable.*

*The study is limited by simulation-based validation and by the need to specify the maximum permissible message age. Nevertheless, the input features and decision criteria do not depend on the services of a particular kernel. Therefore, the method can be adapted for different real-time operating systems and for systems with varying task structures, queue capacities, and workload dynamics.*

*Keywords: real-time operating system, message queue, cascading overload, graphs, exponential smoothing, Monte Carlo method, deadline.*

Стаття надійшла до редакції / Received 30.05.2026
Прийнята до друку / Accepted 15.07.2026
Опубліковано / Published 10.09.2026



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

У кіберфізичних системах результат часто формується не однією задачею, а послідовністю перетворень: зчитування давача, очищення, розпізнавання, вироблення керувальної команди та передавання виконавчому пристрою. Операційна система реального часу забезпечує планування, обмін повідомленнями й синхронізацію, але часову коректність прикладного ланцюга визначає не лише окрема задача, а сукупність затримок між усіма його ланками [1–3]. Доступні засоби спостереження зазвичай описують локальний стан і не відповідають на питання, чи перетвориться зростання однієї черги на прострочення кінцевого результату через кілька наступних задач.

Каскадне перевантаження виникає тоді, коли тимчасова нестача пропускної здатності на одній ділянці спричиняє накопичення повідомлень, затримує надходження даних на наступні ділянки або блокує

попередні задачі через заповнені вихідні черги. Небезпека полягає в тому, що на момент зародження каскаду жодна черга може ще не досягти встановленого порогу. Реакція лише на 80–90 % заповнення тому може бути запізнілою, особливо коли граничний строк стосується всього ланцюга, а не окремої операції.

Отже, початкове накопичення і кінцеве порушення строку можуть бути рознесені як у часі, так і за різними ділянками ланцюга. Перша черга лише вказує місце, де дисбаланс став спостережуваним, але подальший наслідок визначається здатністю всіх залежних задач прийняти й обробити цей потік. Якщо наступна задача має достатню пропускну здатність, короткочасне збільшення кількості повідомлень може зникнути без порушення строку. Якщо ж на наступних етапах уже існує власне накопичення або час обробки збільшився, той самий локальний сплеск переходить у каскад. Саме тому фактичний рівень заповнення однієї черги не є повною характеристикою небезпеки: для прогнозу потрібно простежити, як поточний стан вплине на залишок маршруту повідомлення.

Статичні методи часової перевірки дають строгі оцінки за відомих меж потоків і обслуговування, проте потребують докладної моделі системи до запуску. Методи раннього виявлення прострочень у графах задач розподіляють наскрізний строк між вузлами й можуть враховувати випадковий час виконання [18, 19], але не зосереджуються на оперативному поширенні фактично виміряного накопичення кількох черг. Засоби простеження ланцюгів повідомлень вимірюють уже отриману затримку [15–17], а не обов'язково прогнозують її розвиток.

Метою роботи є розроблення методу, який за локальними вимірюваннями швидкостей надходження, часу обробки й заповнення черг прогнозує поширення перевантаження вперед по ланцюгу та оцінює ймовірність дотримання наскрізного граничного строку. Метод не вимагає від ядра знати внутрішню логіку задач або найгірший час виконання, але вимагає мінімального часового договору: максимального допустимого віку повідомлення чи його абсолютного граничного моменту.

Науковий внесок полягає у цілісному способі оперативного поєднання п'яти складників: виявлення першої небезпечної черги; розкриття лише залежного від неї локального підграфа; згладжене оцінювання надходження та обслуговування; поширення накопичення по ребрах графа; дворівнева імовірнісна оцінка залишкової затримки.

У такій постановці об'єктом прогнозування є не ізольований стан черги, а майбутня траєкторія проходження повідомлення через залежну частину графа. Це уточнення важливе для систем, у яких окрема черга може залишатися напівпорожньою, тоді як наступна задача вже працює на межі пропускної здатності. Отже, сам факт відсутності переповнення не може розглядатися як доказ безпечності ланцюга.

Траєкторія у цьому разі розуміється як послідовність майбутніх станів черг і задач, через які має пройти контрольне повідомлення. Для кожного етапу важливо не лише поточне накопичення, а й те, скільки повідомлень буде попереду на момент фактичного прибуття контрольного елемента. Через це оцінки обчислюються послідовно: результат попереднього етапу визначає час входу та обсяг надходження для наступного. Така залежність відрізняє ланцюговий прогноз від одночасного знімка всіх черг. Знімок показує поточний стан, тоді як траєкторія відображає очікуване переміщення накопичення вперед і дає змогу зіставити сумарну залишкову затримку з доступним часовим запасом.

Запропонований підхід поєднує детерміновану частину з імовірнісною. Детермінована частина відтворює обмежену місткість черг, порядок передавання та доступний часовий запас. Імовірнісна частина описує мінливість потоку і часу обробки. Таке розділення дає змогу не підміняти гарантійний аналіз статистичним прогнозом, а використовувати прогноз як оперативний індикатор для вибору моменту реагування.

У подальшому викладі під локальністю розуміється не спрощення всієї системи до одного вузла, а обмеження аналізу мінімальним досяжним підграфом, який починається з небезпечної черги і закінчується завершальними задачами. Це зменшує обчислювальну ціну виклику та водночас зберігає причинний зв'язок між джерелом накопичення і можливим простроченням.

Межі локального підграфа визначаються структурою залежностей, а не наперед заданою кількістю сусідніх вузлів. Тому для короткого лінійного ланцюга до нього можуть увійти всі наступні задачі, а для великої системи з кількома незалежними контурами – лише невелика частина загального графа. Вузли, які не можуть отримати результат від небезпечної черги, не впливають на строк контрольного повідомлення і не включаються до поточного розрахунку. Водночас локальність не означає ігнорування завершальної частини шляху: обхід триває до тих задач, на яких оброблення повідомлення вважається завершеним. Завдяки цьому зберігається наскрізний зміст оцінки, але обчислення не витрачаються на незалежні компоненти системи.

## АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Поширеним оперативним рішенням є спостереження за часткою заповнення окремої черги та подання сигналу після перетину встановленого порогу. Такий спосіб простий, має малу обчислювальну вартість і не потребує моделі всієї системи. Проте поріг не враховує вік повідомлення, швидкість зміни накопичення, сповільнення наступних задач і різницю між коротким безпечним сплеском та початком

каскаду. Локальне значення також не зберігає причинно-наслідкового зв'язку між повідомленнями різних черг і не містить наскрізного граничного строку.

Мережеве числення описує потоки кривими надходження, а ресурси – кривими обслуговування, після чого в мінімаксній алгебрі виводяться верхні межі накопичення та затримки [8]. Числення реального часу переносить цей апарат на задачі й обчислювальні ресурси вбудованих систем [9-11]. Імовірнісні розширення послаблюють надмірну песимістичність [12], а сучасні огляди показують наявність зрілих засобів обчислення [13]. Математичне порівняння з аналізом часу відгуку засвідчує, що для деяких класів систем ці підходи дають однакові межі [14]. Їхня перевага – формальні гарантії; недолік для поставленої задачі – потреба у завчасно заданих безпечних межах, складність урахування режимних сплесків і висока ціна перебудови моделі під час роботи.

Аналіз часу відгуку ланцюгів обробки визначає наскрізну затримку за відомої схеми викликів і резервування процесорного часу [15]. Спеціалізовані засоби простеження відновлюють шляхи повідомлень та вимірюють змінну затримку [16, 17]. Вони близькі до запропонованої роботи за увагою до причинно-наслідкових ланцюгів, проте переважно виконують вимірювання та оцінювання після проходження даних.

Яно й Азумі запропонували раннє виявлення прострочень у змішаних часових і подієвих орієнтованих ациклічних графах [18]. Тоба й Азумі надалі врахували змінний час виконання як випадкову величину та знизили песимістичність оцінок [19]. Ці роботи є найближчими за метою, однак виходять із відомої структури графа та наскрізного строку, розподіляють часові обмеження між вузлами й не ставлять у центр фактичну динаміку накопичення у зв'язаних чергах. В табл. 1 наведено недоліки відомих рішень щодо раннього виявлення каскадного перевантаження**.** Варто зазначити, що порівняння охоплює рішення, безпосередньо пов'язані з чергами, наскрізною затримкою та раннім виявленням прострочень; воно не є доказом абсолютної відсутності інших реалізацій.

Таблиця 1

**Недоліки відомих рішень щодо раннього виявлення каскадного перевантаження**

| Підхід | Що забезпечує | Недоліки для поставленої задачі |
|---|---|---|
| Порогове спостереження за чергою | Миттєве заповнення окремої черги | Спрацьовує після значного накопичення; не бачить наступні задачі; поріг не пов'язаний із віком повідомлення |
| Мережеве числення | Строгі межі накопичення і затримки | Потребує кривих надходження та обслуговування; межі часто консервативні; модель переважно готується до запуску |
| Числення реального часу | Складальний аналіз потоків і ресурсів | Потрібні формальні моделі компонентів; складно швидко перебудовувати модель за режимних змін |
| Аналіз ланцюгів повідомлень | Наскрізний час відгуку за відомої схеми | Залежить від докладної моделі викликів і планування; не є локальним прогнозом накопичення черг |
| Засоби простеження ланцюгів | Відновлення шляху й вимірювання затримки | Добре спостерігають фактичний шлях, але переважно не передбачають майбутнє накопичення до появи прострочення |
| Раннє виявлення у графах задач | Розподіл наскрізного строку між вузлами; імовірнісний час виконання | Потрібні відомий граф і строк; динаміка черг та локальне розкриття підграфа не є основним об'єктом |

Отже, прогалина полягає не у відсутності математичних засобів для оцінювання затримки, а у відсутності легкого оперативного механізму, який починає аналіз із фактичного зростання конкретної черги, автоматично обмежує розгляд залежним підграфом і видає імовірнісну оцінку до того, як локальний поріг буде досягнуто.

Порівнюваність підходів у цій роботі визначається спільною точкою запуску. кожен спосіб аналізує ту саму послідовність повідомлень і має сформувати попередження до першого фактичного прострочення. Завдяки цьому порівнюється не лише здатність виявити заповнену чергу, а й корисність сигналу для своєчасного втручання. Така постановка відрізняє дослідження від оцінювання середньої пропускної здатності, де запізніле попередження може помилково виглядати прийнятним.

Водночас жоден із розглянутих способів не є універсально кращим за будь-яких умов. Фіксований поріг має мінімальну ціну і може бути достатнім для систем із великим часовим запасом. Строгі методи аналізу корисні під час проєктування та сертифікації. Запропонований метод орієнтований на проміжний режим експлуатації, коли система вже працює, локальні вимірювання доступні, а повна модель усіх потоків або відсутня, або є надто дорогою для постійного перерахунку.

Таким чином, запропонований метод займає проміжне місце між простим оперативним контролем і повним статичним аналізом. Від порогового контролю він успадковує роботу з поточними вимірюваннями та можливість запуску без глобального перерахунку системи. Від методів наскрізної часової оцінки – необхідність простежити залежну послідовність і порівняти залишкову затримку з часовим обмеженням. Водночас результат має інший статус: це не доведена верхня межа, а прогноз, сформований за спостережуваними швидкостями та їх мінливістю. Тому його доцільно використовувати для раннього інформування й вибору моменту реакції, не підмінюючи ним окрему перевірку жорстких часових гарантій.

Нехай прикладна система під керуванням довільної ОСРЧ містить множину задач і черг, а повідомлення проходять від джерела до завершальної задачі. Для кожної черги під час роботи доступні кількість елементів, моменти успішного надсилання та приймання. Для кожної задачі доступні виміряні інтервали виконання, пов'язані з обробленими повідомленнями. Розробник задає для класу повідомлень максимальний вік D або абсолютний граничний момент. Внутрішній зміст повідомлення й семантика обчислень методу не потрібні.

Потрібно в момент $t$ визначити, чи здатне поточне зростання черги $Q_0$ спричинити каскад накопичення у залежній послідовності та прострочення хоча б одного активного повідомлення. Попередження має бути сформоване до фактичного граничного моменту й до повного заповнення черги. Аналіз повинен охоплювати не весь граф системи, а найменший досяжний підграф від $Q_0$ до завершальних задач.

Метод має розв'язувати дві різні задачі прийняття рішення. Перша – подати ранній ризиковий сигнал, коли частка успішних випадкових продовжень стає нижчою за встановлене значення. Друга – подати критичне попередження, коли жодне з $M$ змодельованих продовжень не вкладається у залишок строку. Розділення рівнів потрібне, бо ранній сигнал природно збільшує кількість хибних спрацьовувань.

Без D метод не може встановити факт майбутнього прострочення: однакова черга з десяти повідомлень може бути безпечною для строку 500 мс і неприйнятною для строку 50 мс. За відсутності часового договору допустима лише допоміжна задача – прогноз часу до заповнення черги або перевірка, чи зможе накопичення зменшитися.

Дослідницька гіпотеза полягає в тому, що локальне поширення накопичення з імовірнісним оцінюванням збільшить повноту раннього виявлення порівняно з порогом 80 %, а критичний рівень зменшить кількість хибних спрацьовувань порівняно з одним лише визначеним прогнозом, зберігаючи корисний запас часу.

Вхідні дані методу поділяються на три групи. Перша група описує стан черги: її місткість, поточне накопичення, кількість успішних надсилань і приймань, а також моменти цих подій. Друга група описує оброблення: тривалість завершених операцій, кількість оброблених повідомлень і зв'язок задачі з вхідною та вихідною чергами. Третя група є часовою: вік контрольного повідомлення, максимальний допустимий вік і поточний запас до граничного моменту.

Ці групи дають змогу застосувати метод до різних реалізацій ОСРЧ без доступу до внутрішніх структур планувальника. Потрібні лише події, які можна отримати на точках журналювання надсилання, приймання та завершення обробки. Якщо конкретна система не підтримує журналювання всіх подій, метод може працювати з неповним набором ознак, але тоді результат має трактуватися як попереджувальна оцінка з підвищеною невизначеністю.

Критерієм небезпеки є не саме заповнення, а поєднання трьох умов: локальна черга вже накопичує повідомлення; швидкість надходження наближається до або перевищує швидкість обробки; доступний часовий запас скорочується швидше, ніж компенсується обслуговуванням. Це дозволяє розрізняти безпечний короткий сплеск і тривалу тенденцію, що має потенціал поширитися вперед

Кожна з трьох умов усуває окремий тип передчасного сигналу. Наявність накопичення відокремлює реальний стан черги від абстрактного співвідношення швидкостей за відсутності повідомлень. Порівняння надходження й обробки показує напрямок зміни: за переваги обслуговування навіть помітно заповнена черга здатна поступово спорожніти. Часовий запас, своєю чергою, пов'язує цей дисбаланс із вимогами конкретного повідомлення. Якщо запас великий, тимчасова черга може залишатися допустимою; якщо він майже вичерпаний, небезпечною може бути навіть невелика кількість елементів. Спільне використання умов не вводить нового показника, а поєднує наявні ознаки стану, темпу та строку в єдину точку запуску прогнозу.

## ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Систему подано орієнтованим двочастковим графом: вершини одного типу відповідають чергам, іншого – задачам (рис.1). Ребро від черги до задачі означає читання, а ребро від задачі до черги – запис. Від проблемної черги виконується обхід уперед до завершальних задач. Побудова може виконуватися під час ініціалізації за реєстраційною таблицею об'єктів або під час виконання за точками журналювання; безпечнішим для сертифікованої системи є явне оголошення зв'язків розробником.

Локальне розкриття виконується лише в напрямку руху повідомлень. Зворотні ребра, які не впливають на проходження поточного контрольного повідомлення, не включаються до поточного виклику. Якщо граф містить розгалуження, кожна гілка зберігається окремо, а результат для повідомлення формується за найгіршою прогнозованою залишковою затримкою. Така операція запобігає усередненню гілок із різними часовими властивостями.

Під час обходу для кожної вершини фіксуються її локальні атрибути, а для кожного ребра – правило передавання результату. Найпростішим є співвідношення один-до-одного, але модель допускає розмноження повідомлень, втрати за політикою переповнення та тимчасове блокування вихідної черги. Отже, граф виконує не лише роль ілюстрації, а й визначає порядок обчислень у прогнозі.

Порядок обходу одночасно задає порядок передавання прогнозованих величин. Для вершини-черги потрібні місткість, поточна кількість елементів і оцінка надходження; для вершини-задачі – оцінка швидкості обробки та правило формування виходу. Після оцінювання обробленої кількості результат переноситься ребром до наступної черги й стає частиною її прогнозованого надходження. Якщо задача має кілька виходів, розрахунок продовжується окремо для кожної досяжної гілки. Якщо кілька шляхів ведуть до завершальних задач, повідомлення оцінюється за гілкою з найбільшою залишковою затримкою, оскільки саме вона першою вичерпує доступний строк. Отже, топологія графа безпосередньо визначає, які локальні оцінки мають бути послідовно узгоджені.

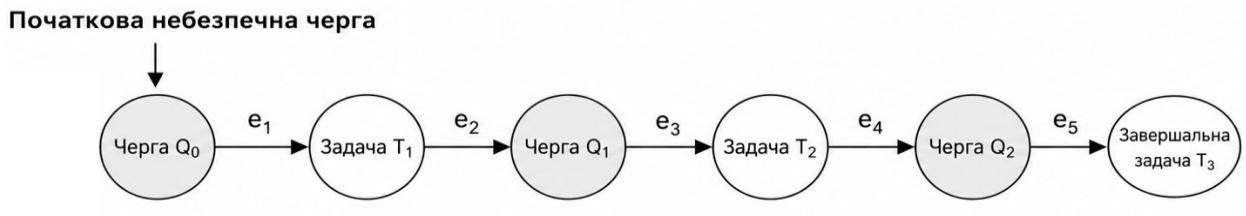


**Рис. 1. Умовний приклад розкриття локального підграфа у напрямку руху повідомлень**

Якщо від проблемної черги не досягається жодної завершальної задачі, прогноз наскрізного дедлайну не формується. У такому випадку метод може повернути локальне попередження про накопичення, але не має права називати його прогнозом прострочення. Це обмеження явно відділяє локальний контроль стану черги від повного ланцюгового прогнозу.

Ця перевірка також запобігає змішуванню двох різних результатів. Локальний сигнал відповідає на питання, чи продовжує конкретна черга накопичувати повідомлення за поточного співвідношення швидкостей. Наскрізний прогноз відповідає на інше питання – чи завершиться оброблення контрольного повідомлення до встановленого моменту. Друге твердження потребує відомого завершального вузла та повного шляху до нього. Якщо шлях не оголошено або його неможливо відновити з журналу, метод не втрачає здатності повідомити про локальний дефіцит, але обмежує зміст попередження. Таке явне розмежування не дозволяє приписувати локальній ознаці сильніший часовий висновок, ніж підтримують доступні дані.

Для формального подання взаємодії черг і задач систему представлено у вигляді орієнтованого двочасткового графа:

$$G = (Q \cup T, E), \tag{1}$$

де $G$ – орієнтований двочастковий граф системи; $Q$ – множина вершин-черг; $T$ – множина вершин-задач; $E$ – множина напрямлених зв'язків; знак розділеного об'єднання означає, що множини $Q$ і $T$ не мають спільних елементів.

Швидкість надходження визначається за кількістю успішно доданих елементів у вікні спостереження. Щоб одиничний випадковий сплеск не спричиняв миттєвого рішення, застосовується експоненційне згладжування, класичний принцип якого сформульовано у [4]:

Вікно спостереження задає часовий масштаб, на якому потік вважається поточним. Кількість надходжень у цьому вікні перетворюється на первинну швидкість, після чого згладжене значення поєднує нове вимірювання з попередньою оцінкою. За більшого коефіцієнта згладжування результат швидше реагує на новий сплеск, але сильніше відображає випадкові коливання. За меншого коефіцієнта зміна проявляється поступово, зате одинична подія менше впливає на рішення. У межах запропонованого методу це значення не використовується окремо: воно надалі зіставляється зі швидкістю обробки, бере участь у попередньому запуску й визначає випадкові надходження під час продовження траєкторії.

$$l_{\text{і}} = \alpha \frac{\Delta N_i}{\Delta t} + (1 - \alpha) l_{i-1}, \tag{2}$$

де $l_{\text{і}}$ – оновлена згладжена швидкість надходження до і-ї черги; $\alpha$ – коефіцієнт згладжування в межах від 0 до 1; $\Delta N_i$ – кількість успішних надходжень у поточному вікні; $\Delta t$ – тривалість вікна; $l_{i-1}$ – попереднє значення швидкості.

Формулу (2) можна деталізувати як два послідовні обчислення. Спочатку кількість успішних надходжень у поточному вікні ділиться на його тривалість, унаслідок чого утворюється незгладжена оцінка швидкості. Потім ця оцінка поєднується з результатом попереднього оновлення. У проміжному записі це має вигляд: $l_{\text{і}}^{(0)} = \Delta N_i \,/\, \Delta t;\;\; l_{\text{і}} = \alpha l_{\text{і}}^{(0)} + (1 - \alpha) l_{i-1}$. Тут $l_i^{(0)}$ позначає лише первинний результат поточного вікна, а не додаткову характеристику методу. Обидва доданки у другому виразі мають однакову розмірність – кількість повідомлень за одиницю часу. Коефіцієнти $\alpha$ та $1 - \alpha$ у сумі дорівнюють одиниці, тому оновлене значення залишається зваженим поєднанням нового вимірювання і попередньої оцінки без зміни одиниць вимірювання.

Походження назви експоненційного згладжування видно після одноразової підстановки попереднього оновлення у формулу (2). Поточне первинне вимірювання отримує вагу α, первинне

вимірювання попереднього вікна – вагу $\alpha(1-\alpha)$, а раніше накопичена оцінка – вагу $(1-\alpha)^2$: $l_i = \alpha l_i^{(0)} + \alpha(1-\alpha)l_{i-1}^{(0)} + (1-\alpha)^2 l_{i-2}$.

Подальші підстановки продовжують той самий ряд: внесок давніших вікон послідовно множиться на $1-\alpha$. Отже, формула не відкидає попередню інформацію миттєво, а поступово зменшує її вплив. Це є розгорнутим поясненням уже використаного у формулі (2) правила і не змінює порядок подальших обчислень.

Пропускна здатність задачі не повинна оцінюватися як кількість вилучень із порожньої черги: відсутність вилучень може означати відсутність попиту, а не зупинку задачі. Тому первинна оцінка будується за фактичними часами обробки повідомлень:

Використання завершених обробок дає змогу відокремити фактичну продуктивність задачі від інтенсивності запитів до черги. Якщо вхідна черга певний час порожня, задача може не виконувати корисної обробки, хоча її здатність обслуговувати повідомлення не погіршилася. Оновлення швидкості нульовим значенням у такій ситуації штучно створило б дефіцит. Тому за відсутності завершених вимірювань попередня оцінка зберігається, а нове значення формується лише тоді, коли відома тривалість фактично виконаної операції. Надалі ця оцінка визначає, яку кількість елементів задача здатна вилучити за один крок і скільки часу контрольне повідомлення очікуватиме за елементами, що перебувають перед ним.

$$m_i = \beta \frac{n_i}{\sum_{j=1}^{n_i} c_{ij}} + (1-\beta)m_{i-1}, \tag{3}$$

де $m_i$ – оновлена згладжена швидкість обробки і-ї задачі; $n_i$ – кількість виміряних обробок у поточному вікні; $c_{ij}$ – тривалість j-ї обробки; $\beta$ – коефіцієнт згладжування; $m_{i-1}$ – попереднє значення швидкості. Якщо $n_i$ дорівнює нулю, значення $m_i$ не оновлюється. Σ

Перший доданок формули (3) також можна отримати через проміжне середнє. Сума тривалостей завершених обробок у вікні дорівнює $C_i = \Sigma_j c_{ij}$, а середня тривалість однієї обробки становить $C_i/n_i$. Швидкість є оберненою до цієї середньої тривалості, тому незгладжене значення послідовно перетворюється так: $\overline{c_i} = (1/n_i)\Sigma_j c_{ij}$; $m_i^{(0)} = 1/\overline{c_i} = n_i/\Sigma_j c_{ij}$.

Після цього до $m_i^{(0)}$ застосовується те саме зважування, що й у формулі (2), але з коефіцієнтом β: $m_i = \beta m_i^{(0)} + (1-\beta)m_{i-1}$. Такий запис показує, чому в чисельнику формули (3) міститься $n_i$, а в знаменнику – загальний час завершених обробок. Якщо у вікні завершено одну операцію, первинна швидкість дорівнює оберненій до її тривалості; якщо завершено кілька операцій, використовується обернена величина до їхньої середньої тривалості.

Окремий випадок $n_i = 0$ не дає можливості обчислити нове середнє, оскільки немає жодної завершеної тривалості. Саме тому в цьому випадку формула не підставляє нульову швидкість, а зберігає $m_{i-1}$. Отже, відсутність нового вимірювання не ототожнюється з відсутністю здатності задачі обробляти повідомлення.

Для випадкових випробувань потрібна не лише середня оцінка, а й мінливість. Розкид оновлюється тим самим потоком вимірювань:

$$v_i = \gamma(x_i - x_{i-1})^2 + (1-\gamma)v_{i-1}, \tag{4}$$

де $v_i$ – оновлена оцінка дисперсії для і-го вузла; $\gamma$ – коефіцієнт оновлення дисперсії; $x_i$ – поточне первинне вимірювання швидкості; $x_{i-1}$ – попередня середня оцінка; $v_{i-1}$ – попередня оцінка дисперсії. Для запуску прогнозування використовується дворівневе правило:

$$x_i \in \{l_i, m_i\}, \tag{5}$$

де $x_i$ – вид швидкості, для якої оновлюється дисперсія; $l_i$ – згладжена швидкість надходження до і-ї черги; $m_i$ – згладжена швидкість обробки і-ї задачі; знак належності вказує, що обчислення виконують окремо для кожної з цих двох величин.

Аналіз не запускається на кожному такті. Спочатку обчислюються відносне заповнення та відносний дефіцит обслуговування. Це обмежує накладні витрати й не активує випадкові випробування для порожніх або стабільних черг:

$$Z_i = 1\left\{\left(\frac{b_i}{K_i} \geq \rho_0 \wedge \frac{l_i - m_i}{\max(m_i, \varepsilon)} \geq \delta\right) \vee \frac{b_i}{K_i} \geq \rho_1\right\}, \tag{6}$$

де $Z_i$ – двійкова ознака запуску прогнозування для і-ї черги; $b_i$ – поточна кількість повідомлень; $K_i$ – місткість черги; $l_i$ – швидкість надходження; $m_i$ – швидкість обробки; ε – мале додатне число для захисту від ділення на нуль; $\rho_0$ – нижній поріг заповнення за наявності дефіциту; $\delta$ – поріг нормованого дефіциту; $\rho_1$ – безумовний вищий поріг; $1\{\cdot\}$ – показникова функція.

У дослідженні використано $\rho_0 = 0{,}18$, $\delta = 0{,}02$ та $\rho_1 = 0{,}32$. Це не універсальні сталі: вони визначають чутливість попереднього запуску, а кінцеве рішення все одно приймається за часовим запасом і ймовірністю.

Дворівневий запуск потрібний для відокремлення помірного, але стійкого дефіциту від різкого наближення до повної місткості. За нижнього рівня заповнення обчислення запускаються лише тоді, коли надходження вже переважає обслуговування на задану нормовану величину. Вищий поріг діє безумовно й не потребує підтвердження дефіцитом, оскільки малий залишок місткості сам по собі вимагає перевірки. Значення порогів у дослідженні задають лише момент переходу від дешевого збору статистики до повного

прогнозу. Вони не визначають остаточний стан повідомлення: після запуску метод усе одно моделює залежний підграф, обчислює часовий запас і оцінює частку успішних продовжень.

Логічний вираз (6) зручно перевіряти через три вже наявні складники: відносне заповнення $q_i = b_i / K_i$, відносну різницю швидкостей $e_i = (l_i - m_i) / max(m_i, \varepsilon)$ та їхній добуток $e_i q_i$. Після такої проміжної заміни правило запуску не змінюється і записується компактніше: $Z_i = 1\{(q_i \geq \rho_0 \wedge e_i q_i \geq \delta) \vee q_i \geq \rho_1\}$.

Послідовність перевірки випливає безпосередньо з цього запису. Якщо $q_i$ досягло $\rho_1$, ознака $Z_i$ дорівнює одиниці незалежно від поточної різниці швидкостей. Якщо $q_i$ ще менше $\rho_1$, але не менше $\rho_0$, потрібне одночасне виконання другої умови – нормований дефіцит з урахуванням фактичного заповнення має досягти $\delta$. За $q_i < \rho_0$ перша частина кон'юнкції є хибною, а безумовний поріг також не досягнуто, тому повний прогноз не запускається. Таким чином, розгорнута перевірка лише пояснює порядок обчислення диз'юнкції та кон'юнкції у формулі (6).

Для кожного дискретного кроку модель визначає, скільки повідомлень може бути оброблено, скільки залишиться у черзі та скільки буде породжено для наступної черги. Така рекурентна форма коректніше відображає кінцеву місткість, ніж необмежене лінійне наростання:

$$s_i = \min\{b_i^- + a_i, m_i \Delta t\}, \tag{7}$$

де $s_i$ – кількість повідомлень, яку і-та задача може вилучити на поточному кроці; $b_i^-$ – накопичення на початку кроку; $a_i$ – нові надходження; $m_i$ – оцінена швидкість обробки; $\Delta t$ – тривалість кроку; $\min\{\cdot\}$ обмежує обробку фактично доступною кількістю повідомлень. Для оновлення поточного накопичення в черзі використовується обмежене рівняння:

$$b_i = \min\{K_i, \max(0, b_i^- + a_i - s_i)\}, \tag{8}$$

де $b_i$ – накопичення наприкінці поточного кроку; $K_i$ – місткість і-ї черги; $b_i^-$ – накопичення на початку кроку; $a_i$ – кількість нових надходжень; $s_i$ – кількість оброблених повідомлень; $\max(0,\cdot)$ забороняє від'ємну кількість елементів, а $\min\{K_i,\cdot\}$ враховує кінцеву місткість. Надлишок понад $K_i$ реєструється як відкидання або блокування надсилання залежно від правил застосунку:

$$a_i = k_{i-1} s_{i-1}, \tag{9}$$

де $a_i$ – надходження до і-ї черги від попереднього етапу; $k_{i-1}$ – середня кількість вихідних повідомлень, породжених попередньою задачею з одного обробленого входу; $s_{i-1}$ – кількість повідомлень, оброблених попередньою задачею. Коефіцієнт $k_{i-1}$ може бути заданий розробником або оцінений за журналом подій.

Формули (7)–(9) слід розглядати як послідовне оновлення одного дискретного кроку. Спочатку визначається доступна для обробки кількість з урахуванням елементів, які вже перебували в черзі, та нових надходжень. Далі залишок обмежується знизу нулем, а згори фактичною місткістю. Після цього оброблена кількість перетворюється на вхід наступної черги відповідно до коефіцієнта породження повідомлень. Тому переповнення на одному етапі не просто фіксується як локальний факт: блокування або відкидання змінює обсяг, який може перейти далі, а затримка готового результату впливає на роботу попередньої задачі. Саме це покрокове узгодження відтворює поширення накопичення вздовж графа.

Проміжні величини для формул (7)–(9) можна розписати без зміни рекурентного правила. На початку кроку доступна кількість повідомлень дорівнює $b_i^- + a_i$, а потенційна кількість оброблень за тривалість $\Delta t$ дорівнює $m_i \Delta t$. Формула (7) вибирає меншу з цих двох величин. Після віднімання обробленої кількості утворюється необмежений залишок $b_i^- + a_i - s_i$, який формула (8) спочатку обмежує знизу нулем, а потім згори місткістю $K_i$: $s_i = min\{b_i^- + a_i, m_i \Delta t\}$; $b_i = min\{K_i, max(0, b_i^- + a_i - s_i)\}$. Якщо $m_i \Delta t$ не менше за $b_i^- + a_i$, задача здатна обробити всі доступні повідомлення на цьому кроці: $s_i = b_i^- + a$, тому внутрішній вираз для залишку дорівнює нулю. Якщо ж $m_i \Delta t$ менше за доступну кількість, обробляється лише $m_i \Delta t$, а різниця переходить у залишок. Доки цей залишок не перевищує $K_i$, він повністю зберігається у черзі; після перевищення місткості формула фіксує $b_i = K_i$, а подальша поведінка надлишку визначається вже зазначеним правилом відкидання або блокування.

Лише після визначення $s_i$формула (9) формує надходження наступного етапу. За $k_i = 1$ кількість переданих далі повідомлень дорівнює кількості оброблених; за значенням, більшим за одиницю, одна обробка в середньому породжує кілька виходів; за значенням, меншим за одиницю, не кожна обробка створює окреме вихідне повідомлення. Отже, порядок (7) → (8) → (9) не можна переставляти: вхід наступної черги залежить від фактично виконаної, а не лише потенційно доступної обробки попередньої задачі.

Оцінка очікування у сталому режимі пов'язана із законом Літтла [5]. У цій роботі використовується не безпосередня стаціонарна формула, а оперативна оцінка часу спорожнення повідомлень перед контрольним елементом. Для контрольного повідомлення рекурсивно враховується накопичення, яке буде наявним у наступній черзі до моменту його прибуття:

$$l_m = \sum_{i=s}^{n} \frac{\tilde{b}_i(\tau_i)+1}{\tilde{m}_i}, \quad (10)$$

де $l_m$– прогнозована залишкова затримка m-го контрольного повідомлення; s – номер початкового етапу локального підграфа; n – номер завершального етапу; $\tau_i$ – прогнозований момент прибуття повідомлення до і-го етапу, послідовно отриманий із попередніх доданків; $\tilde{b}_i(\tau_i)$ – прогнозована кількість повідомлень попереду в цей момент; $\tilde{m}_i$ – прогнозована швидкість обробки; одиниця у чисельнику відповідає самому контрольному повідомленню.

Рекурсивність формули (10) можна показати через моменти переходу між етапами. Для початкової черги момент оцінювання береться рівним поточному часу. На кожному етапі до нього додається час обслуговування прогнозованої кількості повідомлень попереду разом із самим контрольним повідомленням. Після цього отриманий момент використовується для оцінки накопичення вже у наступній черзі: $\tau_s = t$; $\Delta\tau_i = (\tilde{b}_i(\tau_i) + 1) / \tilde{m}_i$; $\tau_{i+1} = \tau_i + \Delta\tau_i$; $l_m = \Sigma_{i=s}^{n} \Delta\tau_i$. Спочатку обчислюється $\Delta\tau_s$ і відповідний момент $\tau_{s+1}$. Далі накопичення $\tilde{b}_{s+1}$ оцінюється не в початковий момент t, а в прогнозований момент фактичного прибуття $\tau_{s+1}$. Та сама операція повторюється до завершального етапу, а сума всіх $\Delta\tau_i$ утворює $l_m$. Таким чином, кожний наступний доданок залежить від попередніх через момент прибуття; формула не підсумовує незалежні поточні знімки черг, а послідовно переносить контрольне повідомлення вздовж уже визначеного підграфа.

Визначимо залишковий часовий запас повідомлення:

$$b_m = D_m - (t - t_m), \quad (11)$$

де $b_m$ – залишковий часовий запас m-го повідомлення; $D_m$ – максимально допустимий вік; t – поточний момент; $t_m$ – час входу повідомлення у ланцюг; різниця $t - t_m$ є його поточним віком. Відношення прогнозованої затримки до наявного часового запасу визначається виразом:

$$r_m = \frac{l_m}{b_m}, \quad (12)$$

де $r_m$ – відношення прогнозованої залишкової затримки до наявного запасу; $l_m$ – прогнозована залишкова затримка m-го повідомлення; $b_m$ – його часовий запас. Якщо $r_m$ не менше одиниці, визначений прогноз не вкладається у строк і формується попередження.

Відношення у формулі (12) нормує прогнозовану затримку відносно запасу конкретного повідомлення. Однакова залишкова затримка має різний зміст для повідомлень різного віку: за великого запасу вона може бути допустимою, а безпосередньо перед граничним моментом – критичною. Значення менше одиниці означає, що середній визначений прогноз ще міститься у доступному інтервалі; значення, яке дорівнює або перевищує одиницю, вказує на прогнозоване вичерпання строку.

Однак ця перевірка спирається на згладжені середні величини й не показує можливого розкиду траєкторій. Тому вона використовується як окремий порівнюваний спосіб і як основа для переходу до імовірнісного продовження, а не як абсолютна гарантія результату.

За додатного залишкового запасу умова $r_m \geq 1$ з формули (12) безпосередньо зводиться до порівняння прогнозованої затримки із цим запасом. Після підстановки формули (11) маємо таку послідовність рівносильних перетворень: $r_m \geq 1 \Leftrightarrow l_m \geq b_m \Leftrightarrow l_m \geq D_m - (t - t_m) \Leftrightarrow (t - t_m) + l_m \geq D_m$. Остання нерівність показує той самий критерій через прогнозований вік повідомлення у момент завершення. Доданок $t - t_m$ є вже накопиченим віком, а $l_m$ – оцінкою часу, який ще потрібний для проходження залежної частини ланцюга. Якщо їхня сума досягає максимально допустимого віку $D_m$, визначене продовження не вкладається у строк. Отже, нормоване відношення $r_m$ і пряме порівняння прогнозованого кінцевого віку з $D_m$ дають однакове рішення; відношення використовується для подання результату у безрозмірному вигляді.

Визначений прогноз не показує невизначеність вимірювань. Для цього застосовано метод Монте-Карло [6]. Потік може перебувати у трьох режимах: нормальному, сплесковому та перевантаженому. Послідовність режимів описано марковським ланцюгом; подібні моделі потоків надходження узагальнено у [7]. Матриця переходів оновлюється за спостережуваними переходами з початковими згладжувальними псевдолічильниками:

$$p_{uv} = \Pr\{Z_{i+1} = v \mid Z_i = u\}, \quad (13)$$

де $p_{uv}$ – ймовірність переходу зі стану u до стану v за один крок; $Z_i$ – поточний режим; $Z_{i+1}$ – режим на наступному кроці; $\Pr\{\cdot\}$ – ймовірність події. Для кожного початкового стану сума ймовірностей переходу до всіх можливих станів дорівнює одиниці.

$$q_i \sim \mathrm{Pois}(l_{i,Z}\Delta t), \quad (14)$$

де $q_i$ – випадкова кількість надходжень до і-ї черги на поточному кроці випробування; $Pois(\cdot)$ – розподіл Пуассона; $l_{i,Z}$ – швидкість надходження до і-ї черги у режимі $Z$; $\Delta t$ – тривалість кроку.

$$d_i \sim F_{i,Z}, \quad (15)$$

де $d_i$ – випадкова тривалість обробки на і-му етапі поточного випробування; $F_{i,Z}$ – оцінений розподіл часу обробки і-ї задачі у режимі $Z$. Оцінимо ймовірність вкластися у строк методом Монте-Карло:

$$p_m = \frac{1}{M}\sum_{j=1}^{M} 1\left\{l_m^{(j)} \le b_m\right\}, \quad (16)$$

де $p_m$ – оцінка ймовірності вкластися у строк для m-го повідомлення; $M$ – кількість випадкових випробувань; $j$ – номер випробування; $l_m^{(j)}$ – залишкова затримка у j-му продовженні; $b_m$ – часовий запас; $1\{\cdot\}$ дорівнює одиниці для успішного продовження і нулю інакше. Для класифікації стану повідомлення використовується таке правило:

$$s_m = \begin{cases} \text{безпечний}, & p_m \ge \theta_r, \\ \text{ризиковий}, & 0 < p_m < \theta_r, \\ \text{критичний}, & p_m = 0. \end{cases} \quad (17)$$

де $s_m$ – стан m-го повідомлення; $\theta_r$ – межа ризикового попередження, у дослідженні 0,65; $p_m$– оцінена ймовірність успіху. Критичний стан за нульового $p_m$ означає відсутність успіхів у вибірці, а не логічний доказ абсолютної неможливості.

Три стани відповідають різній силі наявних свідчень. Безпечний стан означає, що оцінена частка своєчасних завершень не нижча за встановлену ризикову межу. Ризиковий стан з'являється, коли успішні продовження ще існують, але їх частка вже недостатня для впевненого продовження без підготовчих дій. Критичний стан фіксується за відсутності хоча б одного успішного продовження у вибірці з M випробувань. Це розділення дозволяє не прирівнювати перше зниження ймовірності до неминучого прострочення. Воно також пояснює різний зміст реакції: ризиковий рівень призначений для завчасного інформування, тоді як критичний підтверджує значно сильнішу спостережувану небезпеку.

Формулу (16) можна інтерпретувати через цілу кількість успішних продовжень. Нехай $S_m$ дорівнює сумі показникових функцій, тобто кількості випробувань, у яких $l_m{}^{(j)} \le b_m$. Тоді оцінка ймовірності є просто часткою цієї кількості у загальній кількості випробувань: $S_m = \sum_{j=1}^{M} 1\{l_m{}^{(j)} \le b_m\}$; $p_m = S_m / M$. Для використаних $M = 250$ і $\theta_r = 0{,}65$ межа безпечного стану відповідає щонайменше 163 успішним продовженням, оскільки $0{,}65 \cdot 250 = 162{,}5$, а $S_m$ може бути лише цілим числом. Отже, значення $S_m$ від 163 до 250 дають $p_m \ge 0{,}65$; значення від 1 до 162 дають $0 < p_m < 0{,}65$ і відповідають ризиковому стану; $S_m = 0$ дає $p_m = 0$ і відповідає критичному стану. Це лише числове розгортання правила (17) для параметрів проведеного дослідження.

Для правильного тлумачення нульового результату використовується верхня однобічна довірча межа. За незалежних випробувань і нуля успіхів вона має вигляд:

$$u = 1 - 0.05^{1/M}, \quad (18)$$

де $u$ – верхня довірча межа 95 відсотків істинної ймовірності успіху за нульової кількості успіхів; $M$ – кількість випробувань; 0,05 – рівень значущості. Для $M$, що дорівнює 250, межа становить приблизно 1,19 %, тому критичне попередження означає дуже малу, але не математично нульову, можливість вкластися у строк.

Довірча межа потрібна тому, що результат Монте-Карло завжди залежить від скінченної кількості випробувань. Нуль успішних продовжень показує, що успіх не спостерігався в конкретній вибірці, але не дозволяє стверджувати, що його істинна ймовірність дорівнює нулю. За M = 250 наведена межа кількісно уточнює силу критичного сигналу: сумісною з результатом залишається лише мала ймовірність своєчасного завершення. Зі збільшенням M така невизначеність зменшується, але пропорційно зростає обсяг моделювання. Отже, кількість випробувань одночасно впливає на статистичну переконливість попередження та на обчислювальну вартість повного виклику.

Число 1,19% у поясненні до формули (18) отримується прямою підстановкою $M = 250$. Степінь зручно обчислити через натуральний логарифм, не змінюючи самого виразу: $u = 1 - 0.05^{1/250} = 1 - exp(ln(0{,}05)/250) \approx 0{,}01191 = 1{,}19\,\%$. Отримане значення є верхньою межею, сумісною з нулем спостережених успіхів у 250 незалежних випробуваннях на обраному 95-відсотковому рівні. Воно не замінює $p_m$ у класифікації і не додає нового стану, а лише кількісно пояснює, що критичне значення $p_m = 0$ у скінченній вибірці не слід ототожнювати з доведеною нульовою імовірністю успіху.

Якщо застосунок дозволяє керувальні дії, кожна дія задається зміною параметрів: тимчасове підвищення швидкості обробки, обмеження входу, відкидання застарілих повідомлень або спрощення обчислень. Метод повторює оцінювання для допустимих дій і шукає найменш витратну, що повертає ймовірність вище ризикової межі. Вибір керувальної дії виконується за таким критерієм:

$$a = \arg \min_{\substack{x \in A \\ p_m(x) \ge \theta_r}} C(x), \quad (19)$$

де $a$ – вибрана керувальна дія; $x$ – окрема допустима дія; $A$ – множина всіх дозволених дій; $p_m(x)$ – оцінка ймовірності своєчасного завершення після застосування дії $x$; $\theta_r$ – ризикова межа; $C(x)$ – вартість або шкода дії. Якщо жодна дія не задовольняє обмеження за ймовірністю, відновлювальної дії не обирають.

Збір подій виконується постійно з малою вартістю. Розкриття підграфа і випадкові випробування запускаються лише після появи одиничної ознаки $Z_i$. В ОСРЧ доцільно виконувати їх у службовій задачі з нижчим пріоритетом, передаючи їй знімок стану без блокування критичних задач. Послідовність збирання даних, прогнозування та формування попереджень наведено на рис. 2.

Повний цикл методу тому складається з послідовних рівнів, кожен з яких виконується лише за наявності даних попереднього. Спочатку точки журналювання оновлюють лічильники й часові оцінки. Потім перевіряється дешева ознака запуску для окремої черги. За її появи визначається досяжний підграф, для якого виконується визначене поширення накопичення та порівняння із залишком строку. Після цього випадкові продовження враховують мінливість надходження, обробки й переходів між режимами. Завершальним результатом є безпечний, ризиковий або критичний стан разом із оцінкою ймовірності. Така послідовність пояснює, чому ресурсомістка частина не виконується для кожної стабільної черги на кожному такті системи.


**Рис. 2. Послідовність оцінювання та формування попереджень**

Практичний алгоритм можна реалізувати як періодичну службову процедуру або як обробник події зміни стану черги. У першому варіанті процедура запускається з фіксованим періодом і має передбачувану верхню межу вартості. У другому варіанті обчислення виконуються лише після надсилання, приймання або завершення обробки, що зменшує середнє навантаження, але потребує захисту від повторного входу та серії однакових подій.

Перед запуском Монте-Карло службова процедура перевіряє актуальність вимірювань. Якщо вік останнього вимірювання перевищує вік контрольного вікна або в журналі є пропуск подій, прогноз позначається як неповний. Така перевірка не створює штучної впевненості за відсутності даних: система може сформувати попередження, але разом із ним повинна зберегти ознаку зниженої якості спостереження.

Для запобігання коливанню станів між сусідніми викликами доцільно застосовувати гістерезис. Перехід від безпечного стану до ризикового виконується за нижньою межею імовірності, а повернення до безпечного – лише після перевищення вищої межі або після кількох послідовних успішних перевірок. Це не змінює формул прогнозу, але зменшує кількість взаємних перемикань сигналізації в умовах шумного потоку.

З погляду інтеграції з ОСРЧ результат прогнозу краще передавати як подію стану, а не як команду безпосереднього втручання. Службова задача може записати рівень попередження в окрему чергу діагностики, оновити лічильник або передати повідомлення диспетчеру. Автоматичне обмеження входу, відкидання застарілих даних чи зміна пріоритету повинні вмикатися лише політикою конкретного застосунку.

Відокремлення прогнозу від керувальної політики зберігає придатність методу для різних застосунків. Одна й та сама оцінка ризику може вимагати різних дій залежно від цінності повідомлень і допустимих втрат: в одному випадку достатньо запису в журнал, в іншому дозволене обмеження входу, а в третьому будь-яке відкидання заборонене. Метод визначає момент і рівень небезпеки, але не надає діям універсальної семантики. Навіть критерій (19) застосовується лише до множини дій, які розробник заздалегідь визнав допустимими та оцінив за вартістю. Якщо такої множини немає, прогноз залишається інформаційним і не змінює роботу задач або черг.

Результат є інформуючим, доки розробник окремо не дозволить автоматичні дії. Часова складність повного виклику визначається обходом локального підграфа та моделюванням усіх його вузлів у кожному випадковому випробуванні протягом заданої кількості часових кроків:

$$c = O(|V_s| + Mh|V_s|), \quad (20)$$

де $c$ – часова складність одного повного виклику; $|V_s|$ – кількість вузлів локального підграфа; $M$ – кількість випадкових випробувань; $h$ – кількість часових кроків у межах запасу контрольного повідомлення, отримана діленням цього запасу на тривалість кроку з округленням угору. Перший доданок відповідає обходу підграфа, другий – послідовному моделюванню його вузлів у всіх випробуваннях.

Вираз складності (20) можна винести за спільний множник $|V_s|$. Це показує, що і разовий обхід, і повторюване моделювання залежать від того самого локального, а не глобального набору вузлів: $c = O(|V_s| + Mh|V_s|) = O((1 + Mh)|V_s|)$.

Для наведеного нижче розрахунку з $M = 250$ і п’ятьма задачами один часовий крок усіх випадкових продовжень містить 250 · 5 = 1250 оновлень, що збігається з указаним обсягом. Якщо в межах залишкового запасу потрібно h кроків, повторювана частина містить $1250h$ таких оновлень, а разовий обхід локального підграфа додається один раз на початку виклику. Це розгортання не змінює асимптотичної оцінки, а лише показує, як із параметрів M, h та $|V_s|$ утворюється фактична кількість повторень основної операції.

Для вбудованої реалізації достатньо зберігати для кожного вузла середні швидкості, дисперсії, місткість, поточне накопичення та індекси сусідів. За 32-бітних чисел це становить десятки байтів на вузол. Точні витрати залежать від подання графа, генератора випадкових чисел і того, чи обчислюються випробування послідовно або групами.

Складність повного виклику визначається не загальною кількістю об’єктів ОСРЧ, а розміром розкритого підграфа, кількістю випробувань і числом часових кроків у доступному запасі. Обхід графа виконується один раз для поточного виклику, тоді як оновлення станів повторюється для кожного випадкового продовження. Тому скорочення локального підграфа безпосередньо зменшує як обсяг робочих даних, так і кількість повторюваних операцій. Послідовне виконання випробувань дозволяє повторно використовувати один буфер стану, не зберігаючи всі траєкторії одночасно. Це не змінює розраховану частку успіхів, оскільки для кінцевої оцінки достатньо накопичувати кількість випробувань, що вклалися у строк.

**Реалізаційні вимоги та відтворюваність**. Для вбудованої реалізації метод можна розділити на постійний контур збору подій і періодичний контур прогнозування. Перший контур не виконує обходу графа та не генерує випадкових чисел: він лише збільшує лічильники, записує часові мітки й оновлює короткі статистики. Другий контур читає узгоджений знімок цих даних, формує локальний підграф і виконує розрахунок. Розділення зменшує вплив прогнозу на критичні задачі.

Узгоджений знімок потрібен для того, щоб під час одного прогнозу не змішувалися вимірювання різних моментів. У простій однопроцесорній системі це може бути коротке блокування або подвійна буферизація структур стану. Критична секція має охоплювати лише копіювання агрегованих полів, а не сам обхід графа та не випадкове моделювання. Тоді затримка, внесена збором даних, залишається обмеженою.

Обсяг пам’яті визначається кількістю вузлів підграфа, кількістю режимів і розміром статистики для кожної швидкості. Окрім середнього значення, потрібна дисперсія або інша компактна характеристика розкиду. Для систем із жорстким обмеженням пам’яті можна зберігати не всю історію, а лише кілька підсумкових моментів і лічильники переходів між режимами. Це зберігає ідею методу без буфера сирих подій.

Відтворення експерименту потребує фіксації генератора випадкових чисел, параметрів вікон, початкових станів черг, правил втрати повідомлень і критерію прострочення. Без цього однакові назви сценаріїв не гарантують однакової траєкторії. У роботі контрольна траєкторія відокремлена від випадкових випробувань прогнозувача, тому збільшення кількості випробувань не змінює саму подію, яку потрібно виявити.

Такий поділ також визначає межі узагальнення результатів. Отримані показники характеризують запропонований набір сценаріїв і правила їх генерації, а не всі можливі конфігурації FreeRTOS або інших ОСРЧ. Тому під час перенесення на реальний стенд потрібно повторити калібрування параметрів за виміряними потоками та окремо перевірити вплив переривань, блокувань і планувальника.

Відтворюваність у цьому дослідженні стосується не лише повторення підсумкової таблиці, а й відновлення послідовності подій усередині кожного сценарію. Фіксоване початкове число задає однакову контрольну траєкторію, а окремий потік випадкових чисел прогнозувача не повинен змінювати надходження та обробку самої модельованої системи. Завдяки цьому порівняння різної кількості випробувань або різних рівнів попередження відбувається на тих самих небезпечних і безпечних випадках. Параметри вікна, початкові накопичення, місткості та правило переповнення також належать до умов сценарію, оскільки їх зміна створює вже іншу траєкторію й не може вважатися повторенням того самого експерименту.

**Проведення експериментів**. Перевірку проведено як відтворюване дискретно-подієве імітаційне дослідження. Експериментальна модель відтворювала правила черг FreeRTOS: обмежену місткість, послідовне надсилання й приймання повідомлень та утримання готового результату за заповненої вихідної черги [20]. FreeRTOS у цьому розділі є лише випробувальним середовищем; математичні умови методу сформульовано для довільної ОСРЧ. Фізичний мікроконтролер не використовувався, тому результати стосуються якості прогнозу, а не доведених витрат на конкретному пристрої. Випадкову послідовність зафіксовано початковим числом 20260713. Було створено 963 кандидати, після чого сформовано

урівноважений набір із 600 сценаріїв: 300 із фактичним простроченням принаймні одного повідомлення і 300 без прострочення. Основні параметри імітаційного дослідження наведено в табл. 2.

Таблиця 2

**Параметри імітаційного дослідження**

| Параметр | Значення |
|---|---|
| Кількість сценаріїв | 600: 300 із простроченням, 300 без прострочення |
| Кількість задач у ланцюгу | Від 2 до 5 |
| Крок імітації | 5 мс |
| Період оновлення оцінок | 20 мс |
| Нормальна швидкість надходження | 35–75 повідомлень за секунду |
| Множник сплеску | 1,00–3,15 |
| Тривалість сплеску | 120–900 мс |
| Нормальна швидкість обробки | 1,18–1,62 від швидкості надходження |
| Сповільнення під час сплеску | 0,95–1,42 від нормального часу обробки |
| Місткість черг | 18–54 повідомлення |
| Максимальний вік повідомлення | 120–500 мс |
| Коефіцієнти згладжування | $\alpha = \beta = 0{,}25$; $\gamma = 0{,}20$ |
| Кількість випадкових випробувань | M = 250 |
| Межа ризикового рівня | $\theta_r = 0{,}65$ |

Експеримент побудовано як збалансований набір із небезпечних і безпечних сценаріїв. У небезпечних сценаріях хоча б одне повідомлення перевищує допустимий вік, а в безпечних усі повідомлення завершуються в межах заданого строку. Балансування потрібне для того, щоб точність не могла бути отримана простим постійним мовчанням алгоритму: однакова кількість позитивних і негативних випадків робить помітними як пропуски, так і хибні сигнали.

Для кожного сценарію фіксуються чотири моменти: початок контрольної траєкторії, перший сигнал методу, перше фактичне прострочення та завершення моделювання. На цій основі правильним раннім виявленням вважається сигнал, який з'явився до прострочення в небезпечному сценарії. У безпечному сценарії будь-який сигнал вважається хибним спрацьовуванням, оскільки він може призвести до непотрібного обмеження входу або зміни режиму.

Зведений показник використано як компроміс між точністю та повнотою. Він не замінює окремих метрик, а лише полегшує порівняння способів із різною поведінкою. Запас часу вимірюється тільки для правильно виявлених небезпечних сценаріїв; для пропущених випадків він не визначається, оскільки сигналу, від якого можна вести відлік, немає.

Окремо контролюється довжина ланцюга задач. Збільшення кількості етапів одночасно підвищує кількість можливих місць накопичення і збільшує час, потрібний для оброблення одного повідомлення. Тому порівняння способів за різної довжини ланцюга дозволяє перевірити, чи не зводиться перевага методу лише до коротких послідовностей.

У кожному сценарії повідомлення відстежувалося від появи у першій черзі до завершальної задачі. Обслуговування задач моделювалося індивідуальними випадковими тривалостями; за повної вихідної черги задача утримувала готовий результат і тимчасово не брала нове повідомлення. Прострочення фіксувалося у момент вичерпання віку D незалежно від того, була черга повною чи ні. Випадковий потік прогнозувача був відокремлений від потоку самої системи, тому зміна кількості випробувань не змінювала контрольну траєкторію.

Таке відстеження робить одиницею оцінювання не стан системи в довільний момент, а завершення конкретного повідомлення в межах його допустимого віку. Черга може жодного разу не стати повною, але повідомлення все одно буде небезпечним, якщо сумарне очікування та обробка на кількох етапах перевищать D. І навпаки, коротке високе заповнення не обов'язково утворює небезпечний сценарій, якщо накопичення встигає зменшитися й усі повідомлення завершуються своєчасно. Саме ця відмінність пояснює, чому поріг місткості та прогноз строку порівнюються за фактичними простроченнями, а не за кількістю випадків переповнення окремої черги.

Перший спосіб подавав сигнал після досягнення 80 % місткості хоча б однієї черги. Другий використовував згладжені середні швидкості й визначений критерій $r_m$ не менше одиниці. Третій був ризиковим рівнем запропонованого методу зі значенням $p_m$ менше 0,65. Четвертий був критичним рівнем того самого методу: жодне з 250 продовжень не вкладалося у строк.

Попередження в небезпечному сценарії зараховувалося як правильне лише тоді, коли воно виникло раніше за перше фактичне прострочення. Сигнал після прострочення прирівнювався до пропуску. Оцінювалися точність попереджень, повнота виявлення, їхній гармонійний зведений показник і запас часу від першого сигналу до першого прострочення.

За результатати експериментів порівняння способів раннього виявлення за точністю, повнотою, зведеним показником і запасом часу наведено в табл. 3, де ПВ – правильно виявлені небезпечні сценарії; ХС – хибні спрацьовування; ПП – пропущені небезпечні сценарії.

Таблиця 3

**Результати раннього виявлення на 600 сценаріях**

| Спосіб | Кількість випадків | Точність / повнота / зведений, % | Запас, мс |
|---|---|---|---|
| Поріг 80 % | ПВ 122; ХС 2; ПП 178 | 98,4 / 40,7 / 57,5 | 187,3 |
| Визначений прогноз | ПВ 291; ХС 106; ПП 9 | 73,3 / 97,0 / 83,5 | 291,3 |
| Імовірнісний: ризиковий | ПВ 291; ХС 120; ПП 9 | 70,8 / 97,0 / 81,9 | 304,5 |
| Імовірнісний: критичний | ПВ 291; ХС 86; ПП 9 | 77,2 / 97,0 / 86,0 | 264,7 |

Табличні результати показують різні типи помилок. Пороговий спосіб майже не помиляється у випадках, коли подає сигнал, але значна частина небезпечних траєкторій до моменту досягнення 80 % ще не виглядає критичною. Визначений прогноз, навпаки, реагує на дисбаланс швидкостей раніше, тому збільшує повноту, але без імовірнісного розділення режимів подає більше сигналів у безпечних сценаріях.

Порівняння ризикового та критичного рівнів демонструє ціну раннього попередження. Ризиковий рівень забезпечує більший середній запас часу, але допускає більше хибних спрацьовувань. Критичний рівень зберігає ту саму повноту в наведеному наборі й зменшує кількість хибних сигналів, оскільки вимагає сильнішої ознаки неминучого порушення строку. Це обґрунтовує використання двох рівнів замість одного універсального порогу.

Два рівні не є двома незалежними методами: вони використовують ті самі вимірювання, локальний підграф і набір випадкових продовжень, але по-різному інтерпретують кількість успішних результатів. Ризиковий рівень реагує вже на істотне зниження частки успіхів, тому природно з'являється раніше й охоплює більше невизначених випадків. Критичний рівень очікує, доки успішних продовжень у вибірці не залишиться, і завдяки цьому відсіює частину безпечних траєкторій. Отримані показники відображають саме цей компроміс: додатковий запас часу ризикового сигналу супроводжується більшою кількістю хибних спрацьовувань, тоді як критичний сигнал є пізнішим, але точнішим.

Під час інтерпретації відсотків важливо враховувати знаменник. Повнота розраховується від 300 небезпечних сценаріїв, точність – від усіх сценаріїв, для яких спосіб подав сигнал, а запас часу – лише від правильно виявлених небезпечних випадків. Змішування цих знаменників призвело б до хибного висновку, що найраніший спосіб одночасно є і найточнішим.

Результати не дають підстав стверджувати, що метод гарантує відсутність прострочень. Вони показують, що за заданих сценаріїв метод частіше створює корисний запас часу для реагування, ніж фіксований поріг. Гарантія може бути сформована лише окремим аналізом верхніх меж часу виконання та пропускної здатності, який має іншу мету і вхідні припущення [8–14].

Графічне порівняння точності, повноти та зведеного показника для досліджуваних способів наведено на рис. 3

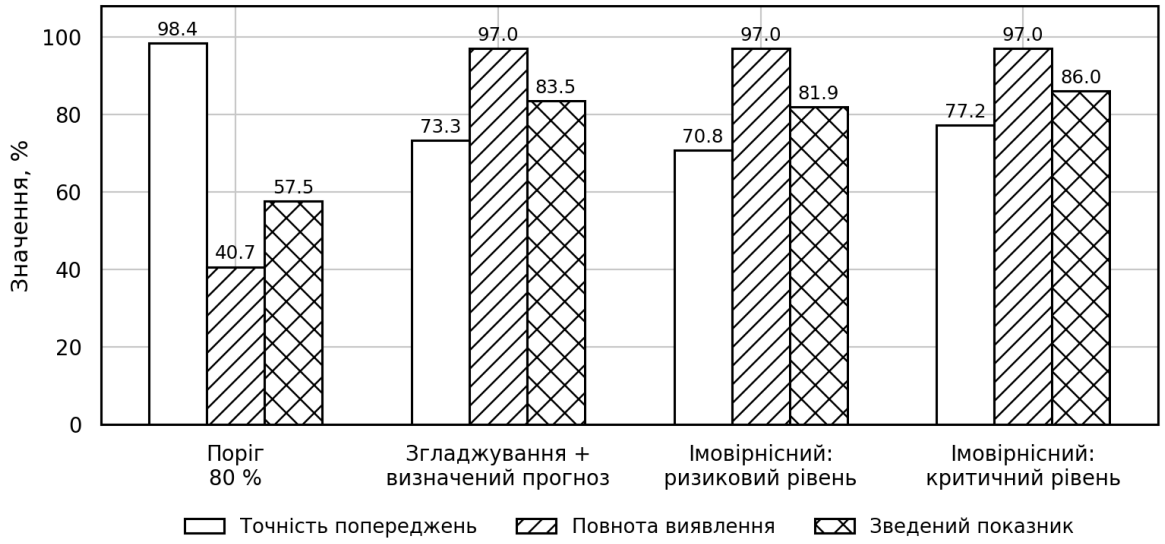


**Рис. 3: Порівняння якості раннього виявлення**

Порогове спостереження майже не помилялося, коли подавало сигнал: точність становила 98,4 %. Проте воно пропустило 178 із 300 небезпечних сценаріїв, тому повнота дорівнювала лише 40,7 %. Це

підтверджує головне припущення роботи: прострочення може виникнути до того, як будь-яка черга досягне 80 % місткості.

Визначений прогноз збільшив повноту до 97,0 %, але спричинив 106 хибних спрацьовувань. Ризиковий рівень подавав сигнал раніше, однак збільшив число хибних спрацьовувань до 120. Критичний рівень зберіг повноту 97,0 % і скоротив їх до 86. Його точність 77,2 % перевищила визначений прогноз на 3,9 відсоткового пункту, а зведений показник 86,0 % – на 2,5 пункту. Порівняно з порогом 80 % повнота критичного рівня зросла на 56,3 пункту.

Числове порівняння також показує, що перевага методу не зводиться до частішого подання будь-якого сигналу. Якби спосіб попереджав майже в усіх сценаріях, висока повнота супроводжувалася б неприйнятною кількістю хибних спрацьовувань. Критичний рівень, зберігши 291 правильно виявлений небезпечний сценарій, зменшив кількість хибних сигналів порівняно з визначеним прогнозом зі 106 до 86. Саме тому разом із повнотою розглядаються точність і гармонійний зведений показник. Вони дають змогу відрізнити реальне покращення балансу помилок від простого зміщення рішення в бік постійного попередження.

Розподіл запасу часу до першого прострочення для кожного способу раннього виявлення наведено на рис. 4.

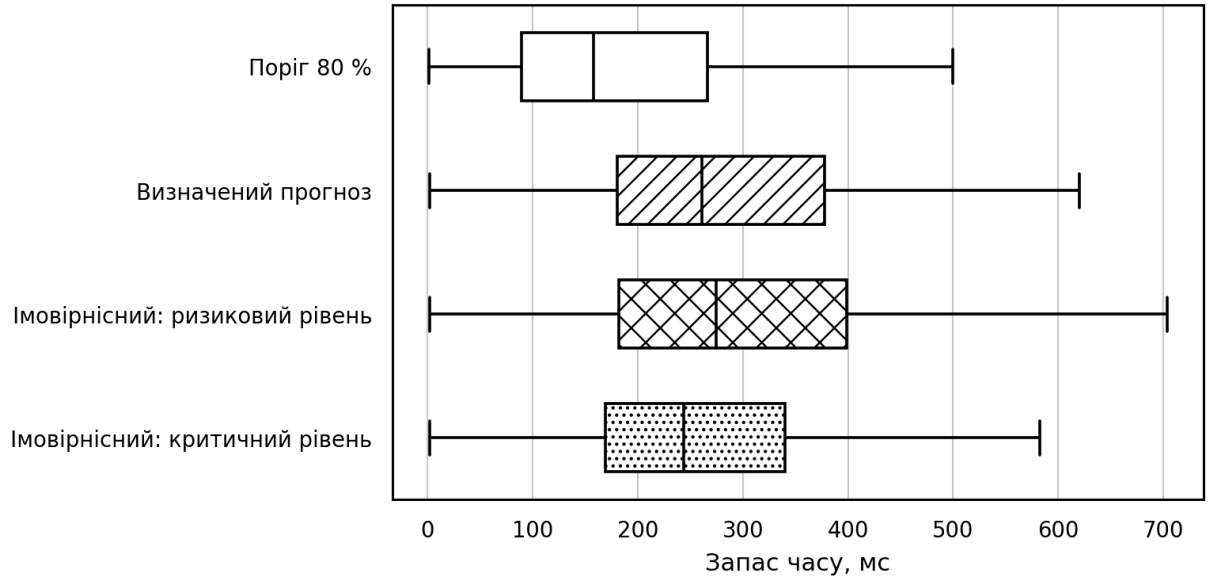


**Рис. 4. Розподіл запасу часу для правильно виявлених небезпечних сценаріїв**

Середній запас часу порогового способу становив 187,3 мс, визначеного прогнозу – 291,3 мс, ризикового рівня – 304,5 мс, критичного – 264,7 мс. Критичний рівень, попри суворішу умову, залишив у середньому на 77,4 мс більше, ніж поріг 80 %. Ризикове попередження випереджало визначений прогноз у середньому на 13,2 мс, але це покращення супроводжувалося зниженням точності на 2,5 відсоткового пункту. Отже, два рівні доцільно використовувати по-різному: ризиковий – для підготовчих дій із малою ціною, критичний – для втручання з відчутними побічними наслідками.

У сценарії № 391 сплеск тривав 709 мс і збільшив надходження у 1,51 раза (рис.5). Ризикове попередження з’явилося на 1200-й мс, визначене – на 1240-й мс, критичне – на 1300-й мс, порогове – на 1380-й мс, а перше прострочення – на 2268,8-й мс. На момент ризикового сигналу найзаповненіша черга ще не досягла порога 80 %. Приклад показує, що джерелом завчасності є не нижчий фіксований поріг, а поєднання темпу зростання, наступних ділянок і залишкового віку повідомлення.

Послідовність сигналів у цьому прикладі узгоджується з логікою чотирьох способів. Ризиковий рівень першим відреагував на зниження частки успішних продовжень. Визначений прогноз сформував сигнал після того, як середня оцінка залишкової затримки перевищила запас. Критичний рівень спрацював пізніше ризикового, коли у вибірці не залишилося своєчасних продовжень. Пороговий контроль очікував фактичного досягнення 80 % місткості й тому виявив небезпеку останнім. Водночас усі сигнали з’явилися до прострочення, а різниця між їх моментами безпосередньо утворила різний запас часу для можливої реакції.

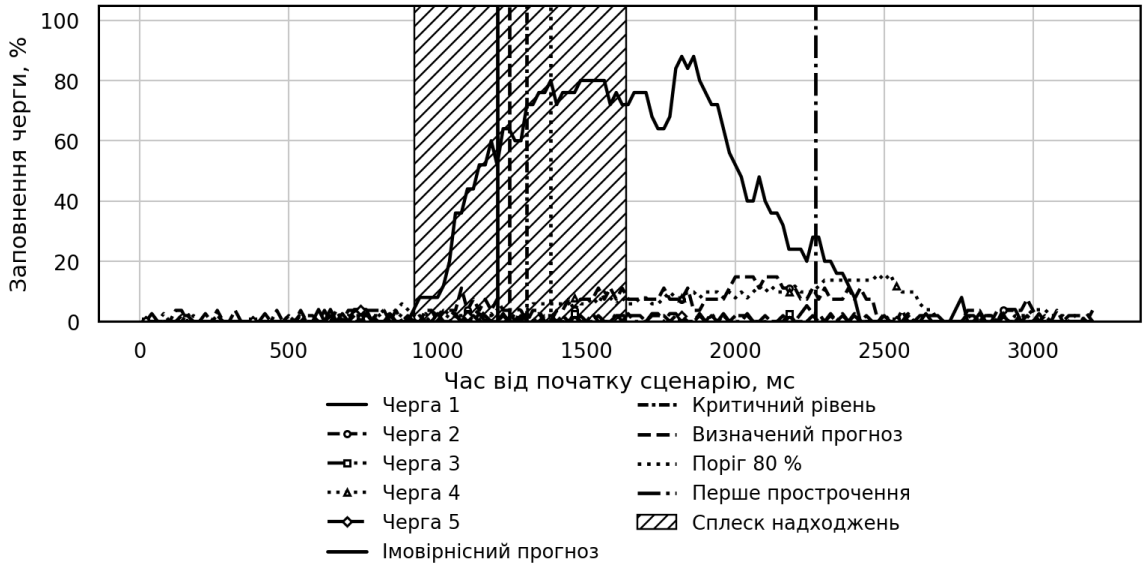


**Рис. 5. Приклад сценарію № 391 із п'ятьма задачами та строком 454 мс**

**Обчислювальні витрати.** Кількість оновлень вузлів зростає лінійно як за M, так і за довжиною локального підграфа. Для п'яти задач і M = 250 потрібно приблизно 1250 оновлень на один крок прогнозу режимів, не враховуючи постійні витрати згладжування (рис. 6).

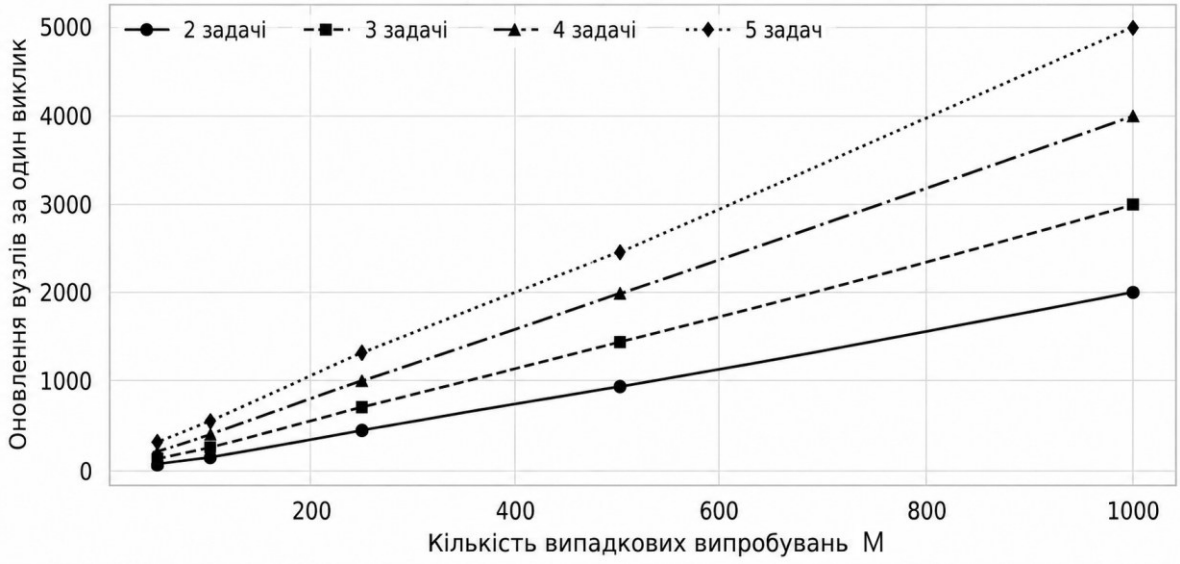


**Рис. 6. Розрахункова залежність обсягу обчислень від M та довжини підграфа**

Довідкова програмна реалізація виконувала повний виклик для п'яти задач і 250 випробувань приблизно за 0,67 мс на обчислювальному вузлі загального призначення. Це значення не можна переносити на мікроконтролер: воно лише підтверджує правильний порядок зростання. Для експерименту з FreeRTOS потрібне окреме вимірювання в тактах процесора, пам'яті та впливу службової задачі на часово критичні задачі. Отримані оцінки стосуються лінійних ланцюгів; розгалуження, злиття та цикли потребують окремої перевірки.

Наведене значення часу слід трактувати разом із теоретичною залежністю на рис. 6. Воно описує один конкретний програмний запуск і не визначає верхньої межі для цільової ОСРЧ. Проте лінійне зростання кількості оновлень пояснює, які параметри потрібно контролювати під час перенесення: розмір локального підграфа, M і кількість кроків до граничного моменту. Збільшення будь-якого з них підвищує деталізацію або статистичну переконливість прогнозу, але водночас збільшує витрати. Тому експеримент на мікроконтролері має окремо виміряти не лише тривалість службового виклику, а й те, чи зберігається передбачуваність виконання основних часово критичних задач.

## ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Запропоновано метод локального прогнозування каскадних перевантажень черг в операційних системах реального часу. Він починає аналіз із небезпечної тенденції конкретної черги, розкриває залежний підграф, згладжує виміряні швидкості надходження й обробки, поширює накопичення по ланцюгу та оцінює дотримання граничного строку за 250 випадковими продовженнями з марковською зміною режимів.

Імітаційне дослідження на 600 сценаріях показало, що поріг 80 % забезпечує високу точність сигналів, але має низьку повноту 40,7 %. Критичний рівень запропонованого методу досяг повноти 97,0 %, точності 77,2 % і зведеного показника 86,0 %, попереджаючи в середньому за 264,7 мс до першого прострочення. Ризиковий рівень збільшив середній запас до 304,5 мс, але подав більше хибних сигналів. Дворівнева схема тому є змістовнішою за один поріг: вона відокремлює ранню підготовку від критичного втручання.

Практична межа методу в тому, що ОСРЧ не може вивести наскрізний строк лише з поведінки черги, тому розробник має задати максимальний допустимий вік повідомлення або його абсолютний граничний момент. Без цього можливий прогноз переповнення, але не висновок про прострочення. Також критичний стан за нуля успішних випробувань є статистичним попередженням; для M = 250 верхня 95-відсоткова межа ймовірності успіху становить приблизно 1,19 %.

Подальша робота буде складати в собі вимірювання на мікроконтролері, перевірку за реальними журналами черг. Окремо потрібне порівняння з повним численням реального часу на однакових наборах, щоб кількісно визначити ціну відмови від жорстких гарантій на користь оперативної пристосовності.

### Література

1. Binqi Sun, Tomasz Kloda, Marco Caccamo, Response Time Analysis for Fixed-Priority Preemptive Uniform Multiprocessor Systems, Proceedings of the 36th Euromicro Conference on Real-Time Systems, Volume 298, 2024, pp. 17:1–17:24, https://doi.org/10.4230/LIPIcs.ECRTS.2024.17.

2. J. M. Aceituno, A. Guasque, P. Balbastre, F. Blanes and L. Pomante, "Optimized Scheduling of Periodic Hard Real-Time Multicore Systems," in IEEE Access, vol. 11, pp. 30027-30039, 2023, https://doi.org/10.1109/ACCESS.2023.3261130.

3. Arvind Easwaran, Michael Yuhas, Saravanan Ramanathan, Ankita Samaddar, Real-Time Scheduling for Computing Architectures, Handbook of Computer Architecture, Springer, 2024, pp. 1–44, https://doi.org/10.1007/978-981-15-6401-7_5-1.

4. Roberts, S. W. (1959). Control Chart Tests Based on Geometric Moving Averages. Technometrics, 1(3), 239–250. https://doi.org/10.1080/00401706.1959.10489860.

5. Little, J. D. C. (1961). A proof for the queuing formula: L = λW. Operations Research, 9(3), 383–387. https://doi.org/10.1287/opre.9.3.383.

6. Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. Journal of the American Statistical Association, 44(247), 335–341. https://doi.org/10.1080/01621459.1949.10483310.

7. Fischer, W., & Meier-Hellstern, K. S. (1993). The Markov-modulated Poisson process cookbook. Performance Evaluation, 18(2), 149–171. https://doi.org/10.1016/0166-5316(93)90035-S.

8. Le Boudec, J.-Y., & Thiran, P. (2001). Network calculus: A theory of deterministic queuing systems for the Internet. Springer, Lecture Notes in Computer Science, 2050. https://doi.org/10.1007/3-540-45318-0.

9. Thiele, L., Chakraborty, S., & Naedele, M. (2000). Real-time calculus for scheduling hard real-time systems. Proceedings of ISCAS 2000, 4, 101–104. https://doi.org/10.1109/ISCAS.2000.858698.

10. Victor Pollex, Frank Slomka, Formal Comparison of Outgoing Event Streams Between Compositional Performance Analysis and Real-Time Calculus, Proceedings of the 37th Euromicro Conference on Real-Time Systems, Volume 335, 2025, pp. 16:1–16:25, https://doi.org/10.4230/LIPIcs.ECRTS.2025.16.

11. Gerlando Sciangula, Daniel Casini, Alessandro Biondi, Claudio Scordino, Marco Di Natale, Bounding the Data-Delivery Latency of DDS Messages in Real-Time Applications, Proceedings of the 35th Euromicro Conference on Real-Time Systems, Volume 262, 2023, pp. 9:1–9:26, https://doi.org/10.4230/LIPIcs.ECRTS.2023.9.

12. Mario Günzel, Niklas Ueter, Kuan-Hsun Chen, Georg von der Brüggen, Jian-Jia Chen, Probabilistic Reaction Time Analysis, ACM Transactions on Embedded Computing Systems, Volume 22, Issue 5s, Article 143, 2023, pp. 143:1–143:22, https://doi.org/10.1145/3609390.

13. Zhou, B., Howenstine, I., Limprapaipong, S., & Cheng, L. (2020). A survey on network calculus tools for network infrastructure in real-time systems. IEEE Access, 8, 223588–223605. https://doi.org/10.1109/ACCESS.2020.3043600.

14. Pollex, V., & Slomka, F. (2022). A mathematical comparison between response-time analysis and real-time calculus for fixed-priority preemptive scheduling. ECRTS 2022, LIPIcs, 231, 7:1–7:25. https://doi.org/10.4230/LIPIcs.ECRTS.2022.7.

15. Casini, D., Blaß, T., Lütkebohle, I., & Brandenburg, B. B. (2019). Response-time analysis of ROS 2 processing chains under reservation-based scheduling. ECRTS 2019, LIPIcs, 133, 6:1–6:23. https://doi.org/10.4230/LIPIcs.ECRTS.2019.6.

16. Kuboichi, T., Hasegawa, A., Peng, B., Miura, K., Funaoka, K., Kato, S., & Azumi, T. (2022). CARET: Chain-aware ROS 2 evaluation tool. IEEE EUC 2022, 1–8. https://doi.org/10.1109/EUC57774.2022.00010.

17. He, X., Sato, H., Okumura, Y., & Azumi, T. (2023). TILDE: Topic-tracking infrastructure for dynamic message latency and deadline evaluator for ROS 2 application. IEEE/ACM DS-RT 2023, 1–9. https://doi.org/10.1109/DS-RT58998.2023.00010.

18. Yano, A., & Azumi, T. (2023). Deadline miss early detection method for mixed timer-driven and event-driven DAG tasks. IEEE Access, 11, 22187–22200. https://doi.org/10.1109/ACCESS.2023.3251359.

19. Toba, H., & Azumi, T. (2024). Deadline miss early detection method for DAG tasks considering variable execution time. ECRTS 2024, LIPIcs, 298, 8:1–8:21. https://doi.org/10.4230/LIPIcs.ECRTS.2024.8.

20. FreeRTOS. (2026). Queue management: API reference. Retrieved July 13, 2026, from https://freertos.org/Documentation/02-Kernel/04-API-references/06-Queues/00-QueueManagement.

## References

1. Binqi Sun, Tomasz Kloda, Marco Caccamo, Response Time Analysis for Fixed-Priority Preemptive Uniform Multiprocessor Systems, Proceedings of the 36th Euromicro Conference on Real-Time Systems, Volume 298, 2024, pp. 17:1–17:24, https://doi.org/10.4230/LIPIcs.ECRTS.2024.17.

2. J. M. Aceituno, A. Guasque, P. Balbastre, F. Blanes and L. Pomante, "Optimized Scheduling of Periodic Hard Real-Time Multicore Systems," in IEEE Access, vol. 11, pp. 30027-30039, 2023, https://doi.org/10.1109/ACCESS.2023.3261130.

3. Arvind Easwaran, Michael Yuhas, Saravanan Ramanathan, Ankita Samaddar, Real-Time Scheduling for Computing Architectures, Handbook of Computer Architecture, Springer, 2024, pp. 1–44, https://doi.org/10.1007/978-981-15-6401-7_5-1.

4. Roberts, S. W. (1959). Control Chart Tests Based on Geometric Moving Averages. Technometrics, 1(3), 239–250. https://doi.org/10.1080/00401706.1959.10489860.

5. Little, J. D. C. (1961). A proof for the queuing formula: L = λW. Operations Research, 9(3), 383–387. https://doi.org/10.1287/opre.9.3.383.

6. Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. Journal of the American Statistical Association, 44(247), 335–341. https://doi.org/10.1080/01621459.1949.10483310.

7. Fischer, W., & Meier-Hellstern, K. S. (1993). The Markov-modulated Poisson process cookbook. Performance Evaluation, 18(2), 149–171. https://doi.org/10.1016/0166-5316(93)90035-S.

8. Le Boudec, J.-Y., & Thiran, P. (2001). Network calculus: A theory of deterministic queuing systems for the Internet. Springer, Lecture Notes in Computer Science, 2050. https://doi.org/10.1007/3-540-45318-0.

9. Thiele, L., Chakraborty, S., & Naedele, M. (2000). Real-time calculus for scheduling hard real-time systems. Proceedings of ISCAS 2000, 4, 101–104. https://doi.org/10.1109/ISCAS.2000.858698.

10. Victor Pollex, Frank Slomka, Formal Comparison of Outgoing Event Streams Between Compositional Performance Analysis and Real-Time Calculus, Proceedings of the 37th Euromicro Conference on Real-Time Systems, Volume 335, 2025, pp. 16:1–16:25, https://doi.org/10.4230/LIPIcs.ECRTS.2025.16.

11. Gerlando Sciangula, Daniel Casini, Alessandro Biondi, Claudio Scordino, Marco Di Natale, Bounding the Data-Delivery Latency of DDS Messages in Real-Time Applications, Proceedings of the 35th Euromicro Conference on Real-Time Systems, Volume 262, 2023, pp. 9:1–9:26, https://doi.org/10.4230/LIPIcs.ECRTS.2023.9.

12. Mario Günzel, Niklas Ueter, Kuan-Hsun Chen, Georg von der Brüggen, Jian-Jia Chen, Probabilistic Reaction Time Analysis, ACM Transactions on Embedded Computing Systems, Volume 22, Issue 5s, Article 143, 2023, pp. 143:1–143:22, https://doi.org/10.1145/3609390.

13. Zhou, B., Howenstine, I., Limprapaipong, S., & Cheng, L. (2020). A survey on network calculus tools for network infrastructure in real-time systems. IEEE Access, 8, 223588–223605. https://doi.org/10.1109/ACCESS.2020.3043600.

14. Pollex, V., & Slomka, F. (2022). A mathematical comparison between response-time analysis and real-time calculus for fixed-priority preemptive scheduling. ECRTS 2022, LIPIcs, 231, 7:1–7:25. https://doi.org/10.4230/LIPIcs.ECRTS.2022.7.

15. Casini, D., Blaß, T., Lütkebohle, I., & Brandenburg, B. B. (2019). Response-time analysis of ROS 2 processing chains under reservation-based scheduling. ECRTS 2019, LIPIcs, 133, 6:1–6:23. https://doi.org/10.4230/LIPIcs.ECRTS.2019.6.

16. Kuboichi, T., Hasegawa, A., Peng, B., Miura, K., Funaoka, K., Kato, S., & Azumi, T. (2022). CARET: Chain-aware ROS 2 evaluation tool. IEEE EUC 2022, 1–8. https://doi.org/10.1109/EUC57774.2022.00010.

17. He, X., Sato, H., Okumura, Y., & Azumi, T. (2023). TILDE: Topic-tracking infrastructure for dynamic message latency and deadline evaluator for ROS 2 application. IEEE/ACM DS-RT 2023, 1–9. https://doi.org/10.1109/DS-RT58998.2023.00010.

18. Yano, A., & Azumi, T. (2023). Deadline miss early detection method for mixed timer-driven and event-driven DAG tasks. IEEE Access, 11, 22187–22200. https://doi.org/10.1109/ACCESS.2023.3251359.

19. Toba, H., & Azumi, T. (2024). Deadline miss early detection method for DAG tasks considering variable execution time. ECRTS 2024, LIPIcs, 298, 8:1–8:21. https://doi.org/10.4230/LIPIcs.ECRTS.2024.8.

20. FreeRTOS. (2026). Queue management: API reference. Retrieved July 13, 2026, from https://freertos.org/Documentation/02-Kernel/04-API-references/06-Queues/00-QueueManagement