

DOI:10.18372/2310-5461.71.21430
УДК 004.75

О. О. Грінченко, к.т.н., доцент,
Національний університет «Київський авіаційний інститут»
<https://orcid.org/0000-0001-9673-6626>
e-mail: gsa_ck@ukr.net;

А. О. Бех, аспірант
Національний університет «Київський авіаційний інститут»
<https://orcid.org/0009-0003-0426-1136>
e-mail: andrii.bekh@gmail.com;

С. А. Грінченко, к.т.н.
Національний транспортний університет
<https://orcid.org/0000-0001-5544-2605>
e-mail: HrinenkoSA@gsuite.duit.edu.ua

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОТОКОЛІВ УПРАВЛІННЯ ТРАНЗАКЦІЯМИ РІС (ОГЛЯД)

Вступ

Стрімкий розвиток розподілених інформаційних систем (РІС) у галузі транспорту зумовлює підвищені вимоги до узгодженості даних, надійності та швидкодії обробки транзакцій. Сучасні транспортні інформаційні системи часто складаються з багатьох взаємодіючих сервісів (модуль бронювання квитків, оплати, логістики тощо), що працюють автономно. Забезпечення коректності транзакцій, які охоплюють декілька таких сервісів, є нетривіальним завданням через необхідність підтримання цілісності даних у розподіленому середовищі [1–2].

Постановка наукової проблеми

У межах цієї роботи під транспортними розподіленими інформаційними системами розуміються системи, що функціонують у межах інформаційно-комунікаційних мереж транспорту, які забезпечують обмін даними між сервісами планування, бронювання, оплати, моніторингу та аналітики. Такі мережі реалізуються на базі стандартних протоколів передачі даних (HTTP, gRPC, MQTT тощо) та архітектурних рішень розподілених обчислень (мікросервіси, хмарні платформи). Фізична транспортна інфраструктура в цьому дослідженні не розглядається, оскільки об'єктом аналізу є інформаційна взаємодія між компонентами цифрових транспортних сервісів [3–4].

Аналіз останніх досліджень і публікацій

Традиційні підходи, орієнтовані на строгі ACID-властивості транзакцій (Atomicity, Consistency, Isolation, Durability – атомарність, узгодже-

ність, ізоляція, довговічність), стикаються з проблемами масштабування та доступності у таких системах. Натомість у розподілених мікросервісних архітектурах набуває популярності концепція BASE (Basically Available, Soft state, Eventual consistency – базова доступність, м'який стан, зрештою узгодженість), що допускає тимчасову неузгодженість заради вищої доступності та продуктивності. В контексті теореми CAP (Consistency, Availability, Partition Tolerance – консистентність, доступність, стійкість до розділення мережі) розробники змушені робити вибір між строгою узгодженістю та доступністю системи у випадку мережних розривів. Як наслідок, у транспортних системах, де критично важлива безперервна робота сервісів, все більшої уваги набувають протоколи, що реалізують компромісні підходи із забезпеченням зрештою досягнутої узгодженості замість повної ACID-консистентності [5–7]. За даними сучасних досліджень [5–8], до найбільш поширених механізмів координації транзакцій у розподілених системах належать класичний двофазний коміт (2PC) та патерн Saga. Також застосовуються їхні варіації та альтернативні моделі, зокрема трифазний коміт (3PC), що розв'язує деякі проблеми блокування 2PC, та модель Try-Confirm/Cancel (TCC), орієнтована на мікросервісні середовища.

Кожен з указаних підходів має свої сильні й слабкі сторони. Протокол 2PC забезпечує глобальну атомарність і строгі гарантії узгодженості даних, але страждає від блокувань і залежності від центрального координатора. Патерн Saga, навпа-

ки, жертвує строгими гарантіями ізоляції заради високої доступності й масштабованості, допускаючи часткову неузгодженість під час виконання довготривалих транзакцій. 2PC додає додаткову фазу узгодження, прагнучи усунути повне блокування учасників при збоях, але ускладнює протокол і висуває вимоги до надійності зв'язку. Модель ТСС спирається на резервування ресурсів та підтвердження або відміну операцій і таким чином забезпечує глобальну узгодженість без жорсткого зв'язування сервісів, хоч і вимагає додаткової логіки у додатках.

Актуальність цього дослідження визначається потребою у виробленні рекомендацій щодо вибору оптимального механізму транзакцій для транспортних інформаційних систем. У таких системах транзакції можуть охоплювати різномірні підсистеми (напрямки перевезень, системи бронювання різних перевізників, платіжні шлюзи тощо), і невдалий вибір протоколу може призвести або до втрати узгодженості даних, або до неприйнятних затримок у обслуговуванні клієнтів.

Мета

Робота спрямована на комплексний аналіз протоколів 2PC, 3PC, Saga та ТСС, зіставлення їх характеристик (узгодженість ACID/BASE, ізоляція,

відмовостійкість, продуктивність) та придатності до використання в транспортних розподілених інформаційних системах.

Виклад основного матеріалу

Під час вибору протоколу для конкретної системи розробникам необхідно враховувати ключові показники продуктивності, зокрема середній час виконання транзакції, максимальну пропускну здатність, а також характер поведінки системи в умовах часткових збоїв, що може проявлятися у вигляді блокування або деградації продуктивності.

Далі використовується поняття раунду R , яке означає один повний цикл обміну повідомленнями між координатором і всіма учасниками та завершується спільним рішенням підтвердження або відкату. Для прикладу, класичний протокол 2PC потребує двох раундів зв'язку, тоді як 3PC – трьох раундів зв'язку. У підході Saga глобальний коміт відсутній, натомість використовується послідовність локальних транзакцій із механізмами компенсації. У моделі ТСС застосовуються два етапи, які можуть виконуватися паралельно для всіх учасників. На основі аналізу літературних джерел [5], [8–10] нижче наведено порівняння протоколів 2PC, 3PC, Saga і ТСС за ключовими метриками (табл. 1).

Таблиця 1

Порівняння протоколів управління транзакціями

Протокол	Ізоляція / узгодженість	R (раунди)	Затримка	Пропускна здатність	Блокування при збої	Складність реалізації
2PC	ACID	2	Помірно-висока	Середня	Так	Низька/Середня
3PC	ACID	3	Висока	Низька	Ні	Висока
Saga	BASE/eventual	0 (глобального коміту немає; n локальних кроків)	Низька на крок	Висока	Ні	Середня/ Висока
ТСС	ACID-подібна	2	Помірна	Висока	Ні	Висока

У таблиці використовується наступна шкала затримки:

- Низька – $R = 0 \dots 1$;
- Помірна – $R=2$;
- Помірно-висока – $R = 2$, але з додатковими комунікаційними витратами
- Висока – $R \geq 3$.

Із наведеної таблиці видно, що протоколи 2PC і 3PC забезпечують сильні гарантії узгодженості ціною зниження продуктивності, тоді як підходи Saga і ТСС орієнтовані на підвищення продуктивності та доступності за рахунок відмови від не-

гайної консистентності або ускладнення прикладної логіки. На практиці вибір конкретного підходу визначається вимогами транспортної інформаційної системи. У випадках, коли критичною є відсутність розбіжностей у даних, зокрема в білетних системах реального часу з централізованим управлінням, застосування 2PC може бути виправданим попри додаткові затримки. Натомість у системах, що інтегрують велику кількість незалежних компонентів і потребують швидкого відгуку, наприклад на платформах спільного використання транспорту, де допустимою є незначна

тимчасова несинхронність даних, більш ефективними є підходи Saga або TCC.

Більш детальний аналіз кожної характеристики наведений нижче.

Затримка в протоколі 2PC зумовлена тим, що до кожної транзакції додається щонайменше два раунди мережевого обміну повідомленнями, а саме фази підготовки та коміту, унаслідок чого латентність є помірно високою. Протокол 3PC містить додаткову фазу, що призводить до ще більшого загального часу підтвердження, тому за показником латентності він є найповільнішим. Патерн Saga зазвичай характеризується низькою затримкою на кожному окремому кроці, оскільки не потребує очікування глобального коміту, однак повна Saga-транзакція може включати декілька послідовних раундів, через що сумарний час виконання зростає зі збільшенням їх кількості. На практиці реалізації Saga часто налаштовуються таким чином, щоб більшість раундів виконувалися асинхронно, що дозволяє приховати частину затримок від кінцевого користувача, наприклад коли підтвердження надсилається вже після відображення успішного завершення замовлення. Підхід TCC характеризується помірною затримкою, оскільки етап Try виконується паралельно для всіх учасників, а подальший етап Confirm також може виконуватися паралельно. У результаті час виконання TCC є близьким до 2PC, що відповідає двом раундам обміну, а в багатьох випадках навіть меншим за умови швидкого виконання резервування [6], [8–11].

Пропускна здатність протоколу 2PC обмежується зниженим рівнем паралелізму, оскільки численні транзакції, що конкурують за спільні ресурси, вимушені чекати завершення одна одної через механізми блокування, унаслідок чого максимальна кількість успішно оброблених транзакцій за секунду є середньою. Протокол 3PC має ще більш жорсткі обмеження, зумовлені більшою тривалістю кожної транзакції та збільшеною кількістю повідомлень, тому його пропускна здатність є найнижчою. Патерн Saga, навпаки, забезпечує дуже високу пропускну здатність, оскільки дозволяє транзакціям виконуватися паралельно та переплітатися, коли одна Saga може виконувати черговий крок у сервісі A, тоді як інша одночасно виконується у сервісі B, а навіть у межах одного сервісу кілька Saga можуть оброблятися послідовно без глобального простою системи. Відсутність глобальних блокувань дає змогу обробляти велику кількість транзакцій одночасно в межах доступних ресурсів окремих сервісів. Підхід TCC також орієнтований на високий рівень паралелізму, оскільки резервування ресурсів не блокує виконання інших транзакцій за умови, що вони

працюють з іншими ресурсами. Певне зменшення пропускну здатності можливе у випадках, коли окремий сервіс отримує значну кількість Try-запитів і тимчасово резервує велику кількість ресурсів, однак зазвичай цей ефект є керованим, наприклад у ситуації черги на популярний рейс, коли всі місця можуть бути зарезервовані й інші клієнти очікують скасування резервів або їх підтвердження [8], [10–11].

Блокування при збоях у протоколі 2PC зумовлене тим, що в разі відмови координатора або втрати повідомлень учасники можуть залишатися в заблокованому стані до моменту відновлення системи. Це є суттєвим недоліком 2PC, оскільки обробка інших операцій, пов'язаних із відповідними ресурсами, може бути вимушено призупинена. Протокол 3PC було розроблено з метою усунення цієї проблеми, і за умови дотримання відповідної моделі відмов, зокрема fail-stop та наявності відомих таймаутів, він є неблокуючим. У разі виходу з ладу координатора або окремого учасника решта вузлів здатні самостійно прийняти узгоджене рішення завдяки використанню фази PreCommit. У результаті, в ідеалізованій моделі, 3PC не залишає систему в підвішеному стані, оскільки транзакція завжди або завершується, або відкочується без необхідності адміністративного втручання [12]. Патерн Saga також не призводить до блокування ресурсів у разі виникнення помилок, оскільки будь-який збій ініціює виконання компенсаційних дій, при цьому паралельна діяльність системи не зупиняється, за винятком можливих локальних компенсацій у відповідь на виявлену неконсистентність. Підхід TCC аналогічно уникає тривалого блокування, оскільки за відсутності підтвердження з боку учасника після спливання таймауту або за сигналом координатора виконується етап Cancel для вже створених резервів. Це означає, що системні ресурси в найгіршому випадку залишаються заблокованими лише тимчасово до моменту завершення таймауту резервування, але не на невизначений строк. Загалом протокол 2PC є єдиним серед розглянутих чотирьох підходів, який за відсутності спеціальних механізмів відновлення може призвести до нескінченного блокування у випадку серйозних збоїв [6], [8–11].

Стійкість до збоїв у даному контексті розуміється як здатність системи довести транзакцію до визначеного результату у вигляді коміту або відкату попри відмови окремих вузлів. Протокол 2PC характеризується обмеженою відмовостійкістю, оскільки відмова координатора в критичний момент залишає результат транзакції невизначеним, тоді як відмова будь-якого учасника фактично призводить до відмови всієї транзакції,

що, однак, є безпечним варіантом, адже система виконує відкат. Протокол 3PC демонструє вищу стійкість, оскільки відмови координатора або учасників враховані в його логіці, і транзакція може бути завершена за участі доступних вузлів, проте це справедливо лише за умови синхронного середовища. Патерн Saga відзначається високою стійкістю до збоїв окремих сервісів, адже у разі недоступності або помилки будь-якого з них виконується ланцюжок компенсацій, після чого транзакція завершується відкатом, при цьому система продовжує функціонувати, а дані залишаються узгодженими, хоча й з певною затримкою. Відсутність централізованої точки відмови означає, що збій окремого сервісу не призводить до паралічу інших процесів. Підхід ТСС демонструє аналогічну стійкість до збоїв учасників, оскільки за відсутності підтвердження з боку окремого сервісу транзакція скасовується, а інші учасники виконують повернення зарезервованих ресурсів. Водночас у цій моделі наявний координатор, роль якого може виконувати ініціатор транзакції або спеціалізований менеджер, і у разі його відмови учасники спираються на механізми самоочищення, зокрема таймаути резервування, однак для коректного завершення транзакцій після перезапуску доцільно використовувати резервування координатора або журналювання станів [5], [10], [12].

Модель узгодженості в протоколах 2PC і 3PC ґрунтується на забезпеченні суворої узгодженості, тобто ACID-серіалізованості, за якої результати транзакції стають видимими для всіх учасників одночасно лише після виконання коміту, а всі

проміжні стани залишаються прихованими. Патерн Saga реалізує модель узгодженості типу зрештою досягнутої узгодженості, відповідно до якої системі може знадобитися певний час після початку виконання саги для досягнення узгодженого стану, особливо у випадках запуску компенсаційних дій, проте зрештою всі вузли погоджуються щодо кінцевого результату. Водночас Saga не гарантує ізоляції транзакцій, що вимагає від системи та прикладної логіки толерантності до тимчасових розбіжностей у даних. Підхід ТСС передбачає наявність проміжного стану типу «зарезервовано», який свідчить про те, що повна консистентність ще не досягнута, але транзакція цілеспрямовано рухається до неї. З точки зору зовнішнього спостерігача модель узгодженості ТСС також є ближчою до зрештою досягнутої узгодженості, хоча у випадку успішного підтвердження фінальний стан стає строго узгодженим для всіх учасників [5], [8], [11].

Розглянуті вище протоколи по-різному застосовуються на практиці (таблиця 2) в транспортній галузі, залежно від специфіки інформаційної системи. Традиційно системи бронювання (наприклад, залізничні або авіаційні) будувалися як централізовані і використовували двофазний коміт у рамках одного підприємства (єдиної бази даних з розподіленими філіями). Сучасні тенденції переходу до мікросервісів та інтеграції різних перевізників вимагають більш гнучких підходів [1–4], [13–14].

Таблиця 2

Огляд застосування протоколів у транспортних РІС

Бізнес функція	Ключові вимоги	Рекомендований підхід	Причина вибору	Умови / обмеження
Бронювання / продаж	Консистентність: кінцева узгодженість Затримка: $\leq 1-2$ с	Saga	Уникає блокувань; підтримує компенсації між сервісами	Ідемпотентність, дедублікація, таймаути; узгоджені бізнес-компенсації
Управління місцями	Консистентність: кінцева узгодженість (з квотою на читання); Затримка: $\leq 500-800$ мс	Saga	Підсилює масштабованість, дозволяє квотувати й компенсувати	Квота-кеш для ізолюваного читання; підтвердження лише наприкінці ланцюжка
Диспетчеризація рейсів / маршрутів	Консистентність: кінцева узгодженість; Затримка: ≤ 1 с	Saga	Краща стійкість до часових збоїв та перепланувань	Кореляційні ID, гарантія порядку подій, дедублікація
Облік багажу / вантажу	Консистентність: кінцева узгодженість; Затримка: ≤ 1 с	Saga	Підтримує ідемпотентність і відновлення при високому потоці	Черга необроблених повідомлень, повторна обробка, уніфіковані коди відкату
Телеметрія та події від ТЗ	Консистентність: кінцева узгодженість; Затримка: мс–с, високий TPS;	Saga	Без глобальних блокувань; гнучка компенсаційна семантика	Відсутність глобального блокування; потрібна телеметрія й моніторинг спроб

Закінчення табл. 2

Бізнес функція	Ключові вимоги	Рекомендований підхід	Причина вибору	Умови / обмеження
Платіж / повернення	Консистентність: повна. Затримка: $\leq 2-3$ с;	TCC	Потрібні холди/резервування з чітким confirm/cancel	Етапи TCC; терміни холдів; ідемпотентність підтверджень
Розрахунок тарифу	Консистентність: моментального знімку/кінцева узгодженість. Затримка: ≤ 500 мс онлайн	Saga	Дозволяє онлайн-операції та відкладений коректний перерахунок	Контроль даних; періодичний перерахунок, гарантований рівень якості на затримку
Підтримка клієнта	Консистентність: кінцева узгодженість/ повна	Saga	Гнучко розв'язує спірні сценарії з компенсаціями	Фіксація фінальних статусів; аудит; за потреби – 2PC у фіналі
Аналітика / звіти в реальному часі	Консистентність: кінцева узгодженість. Затримка: секунди	Saga	Глобальна атомарність не потрібна; важливі швидкі оновлення	Часові мітки, відновлення матеріалізованих представлень; контроль запізнення
Ідентифікація / контроль доступу	Консистентність: повна для політик. Затримка: $\leq 200-300$ мс	2PC	Строгі політики доступу потребують атомарності й узгодженості	Локальні транзакції та централізований журнал; короткі блокування
Інтеграція з сторонніми провайдерами	Консистентність: залежить від гарантованого рівня якості. Затримка: $\leq 1-2$ с	Saga	Стійка до нестабільності API третіх сторін	Спроби / таймаути; контрактні компенсації; ідемпотентні ключі

У системах планування складних маршрутів, що охоплюють кілька видів транспорту та різних операторів, ключовим завданням є узгодження дій незалежних інформаційних систем. Наприклад, платформа, яка агрегує пропозиції авіакомпаній, залізничних операторів і автоперевізників для формування єдиного маршруту, не може примусити всі сторони брати участь у спільній транзакції за протоколом 2PC, оскільки кожен оператор використовує власну систему без спільного координатора. У таких умовах доцільним є застосування патернів Saga та TCC. Зазвичай Saga використовується для оркестрації процесу бронювання, за якого система поетапно резервує окремі сегменти поїздки. У разі недоступності або непідтвердження будь-якої ділянки маршруту раніше заброньовані сегменти скасовуються за допомогою компенсаційних операцій. Подібний підхід реалізовано в багатьох агрегаторах подорожей і він забезпечує користувачу зручність оформлення єдиного замовлення для складної багатосегментної поїздки. Водночас за умови, що перевізники підтримують механізм тимчасового резервування ресурсів, може застосовуватися модель TCC, за якої система надсилає Try-запити кожному оператору, наприклад тимчасово резервуючи місце в літаку та місце в поїзді на кілька хвилин до моменту підтвердження покупки користувачем. Після

цього система або надсилає підтвердження усім учасникам, у результаті чого користувач отримує повний набір квитків, або ініціює відміну і всі резерви знімаються. Така форма координації покращує користувацький досвід завдяки зменшенню кількості невдалих транзакцій, однак вимагає відповідної підтримки з боку кожного оператора.

У міських транспортних системах (наприклад, інтегровані електронні квитки на метро, автобус, трамвай) також виникає потреба розподіленої транзакційності. При поповненні балансу транспортної карти транзакція може включати списання коштів через банківську систему та зарахування по різних підсистемах транспорту. У таких умовах класичний протокол 2PC практично не застосовується через гетерогенність учасників, оскільки банківські та транспортні сервери не мають спільного транзакційного менеджера. Натомість для синхронізації використовується патерн Saga, за якого спочатку виконується платіж у вигляді локальної транзакції в банківському сервісі, а подія успішної оплати ініціює оновлення балансу в інформаційній системі перевізника. У разі виникнення помилки на етапі оплати Saga завершується без подальших дій, тобто кошти не списуються і необхідність в оновленні балансу відсутня. Якщо ж помилка виникла на етапі оновлення балансу (наприклад, збій зв'язку з транспортним

сервером після списання коштів), Saga ініціює компенсацію платежу – автоматичне повернення коштів користувачу. Такий механізм забезпечує модель зрештою досягнутої узгодженості, за якої кошти пасажирів або зараховуються на транспортну карту, або повертаються назад, хоча в окремий короткий момент після виникнення збою вони могли вважатися списаними без відповідного оновлення балансу. Виконання компенсаційної транзакції усуває цю тимчасову невідповідність і відновлює узгоджений стан системи [13].

У системах керування вантажопотоками, які охоплюють декілька взаємодіючих учасників, зокрема склади, перевізників і митні органи, також широко застосовуються розподілені транзакційні шаблони. Зокрема, процес оформлення та відправлення вантажу може бути змодельований у вигляді Saga, де послідовно виконуються такі етапи, як створення замовлення на перевезення, резервування транспортного засобу та оформлення митної документації, при цьому кожен крок підтверджується незалежно. У разі невдалого завершення будь-якого етапу, наприклад митного оформлення, ініціюється виконання компенсаційних операцій, зокрема скасування бронювання транспорту та видалення замовлення. Це дозволяє системі залишатися узгодженою без використання глобальних блокувань, що особливо важливо, коли учасники – окремі організації зі своїми системами. Підхід апробовано в авторських тезах [15].

Загалом у транспортних інформаційних системах спостерігається така закономірність, за якої у випадках, коли всі учасники транзакції перебувають під спільним адміністративним контролем і використовують єдину базу даних або спільний транзакційний координатор, на практиці застосовується протокол 2PC для досягнення суворой ACID-узгодженості. Водночас значні координаційні витрати та ризики блокування обмежують ефективність 2PC у міжорганізаційних або високолатентних сценаріях, що зумовлює його обмежене застосування в таких умовах [6], [8–9]. Поза межами одного домену, особливо у багатосторонніх інтеграціях (агрегатори поїздок, мультимодальні маршрути), переважають підходи з зрештою досягнутої узгодженості - насамперед Saga з компенсаціями або TCC з резервуванням ресурсів [5], [7]. 3PC, попри неблокувальну поведінку в ідеалізованих умовах, рідко обирають у прикладних рішеннях через додаткову фазу, складність та комунікаційні накладні витрати [8–9]. Паралельно, для надійності критичних компонентів часто використовують консенсус усередині СУБД (Raft/Raxos), що вирішує задачі реплікації та узгодження станів, але не підміняє керування приклад-

ними міжсервісними транзакціями [8]. В транспортних PIC це призводить до гібридних стратегій (Saga/TCC для міжорганізаційних ланцюжків і локалізований 2PC у межах одного провайдера), які практично підтверджені в галузевих кейсах і дослідженнях [5–7], [10–11].

Висновки

У даній роботі виконано порівняльний аналіз протоколів управління транзакціями для транспортних розподілених інформаційних систем (2PC, 3PC, Saga, TCC) за критеріями консистентності, продуктивності, масштабованості та відмовостійкості. Показано, що 2PC забезпечує строгі ACID-гарантії й доцільний у сценаріях із критичною цілісністю даних, проте має підвищену латентність, залежність від координатора та гіршу масштабованість [5]. 3PC частково розв'язує проблему блокування, вводячи проміжну фазу узгодження, однак вимагає суворого дотримання припущень про надійність мережі і рідко використовується через підвищену складність [12]. Патерн Saga демонструє вищу пропускну здатність і доступність завдяки використанню локальних транзакцій і компенсаційних операцій, а його ключовий недолік, що полягає у відсутності повної ізоляції, може бути нівельований шляхом застосування семантичних блокувань, ідемпотентних інтерфейсів, шаблону надійної публікації подій із транзакційного сховища, а також архітектурного підходу розділення моделей читання та запису з фіксацією подій [6], [11]. TCC виступає практичним компромісом для сценаріїв «бронювання» ресурсів, але потребує ретельного налаштування таймаутів і компенсацій [10].

На підставі проведеного аналізу сформовано рекомендації щодо вибору підходів, відповідно до яких протокол 2PC доцільно застосовувати для фінансово критичних операцій із жорсткими вимогами до узгодженості, патерн Saga є оптимальним для більшості високонавантажених процесів у транспортних розподілених інформаційних системах, а модель TCC рекомендована для інтеграційних сценаріїв і задач резервування, у яких природно підтримується тимчасове утримання ресурсів. Запропоновані техніки інтеграції повідомлень та контрзаходи проти аномалій ізоляції дозволяють зменшити операційні ризики за умови збереження високого рівня продуктивності.

Отже, поставлена у вступі мета - узагальнити сучасні підходи й надати обґрунтовані критерії вибору протоколів для транспортних PIC - досягнута. Результати можуть слугувати методичною основою для проєктування та еволюції систем і окреслюють подальші напрями досліджень (гіб-

ридні моделі, адаптивні політики ізоляції, формальна верифікація компенсацій).

ЛІТЕРАТУРА

- [1] Довженко Н. М., Мазур Н. П., Костюк Ю. В., Рзаєва С. Л. Інтеграція IoT та III в інтелектуальні транспортні системи. *Кибербезпека: освіта, наука, техніка*. 2024. DOI: 10.28925/2663-4023.2024.26. 708.
- [2] Кучук Н. Г., Шиман А. П. Інтелектуальні транспортні системи : навчальні матеріали. Харків : НТУ «ХПІ», 2023–2024. URL: <https://repository.kpi.kharkov.ua/bitstreams/7fcf6869-a110-469b-9535-0035ee8d91b3/>
- [3] Тимошук О. М., Мельник О. В. Інформаційно-логістичні системи в сучасних транспортних технологіях. *Інвестиції: практика та досвід*. 2015. № 22. С. 79–82. URL: <http://www.investplan.com.ua/?op=1&z=4729&i=>
- [4] Мурований І. Інформаційні системи в сучасних транспортних технологіях. Проблеми з транспортними потоками і напрямки їх розв'язання : тези доповідей. Львів : Національний університет «Львівська політехніка», 2019. С. 54–55. URL: <https://ena.lpnu.ua/items/9227d50c-9f6b-42f9-9074-378f1ede1613>
- [5] Дарагмі Е., Чжан Ц.-П., Юань С.-М. Удосконалення патерну Saga для розподілених транзакцій у мікросервісній архітектурі. *Прикладні науки*. 2022. Т. 12, № 12. С. 6242. DOI: 10.3390/app12126242.
- [6] Талавер О. В., Вакалюк Т. А. Надійні розподілені системи: огляд сучасних підходів. *Журнал периферійних обчислень*. 2023. Т. 2, № 1. С. 84–101. DOI: 10.55056/jec.586.
- [7] Бех А. О. Огляд сучасних протоколів управління транзакціями. Інженерія програмного забезпечення – 2025 : матеріали Міжнар. наук.-практ. конф. Київ, 2025. С. 6–8.
- [8] Баштовий А., Фечан А. Розподілені транзакції в мікросервісній архітектурі: стратегії обґрунтованого вибору. *Інформаційні системи та мережі*. 2024. № 15. С. 449–463. DOI: 10.23939/sisn2024.15.449.
- [9] Хеббар К. С. Оптимізація розподілених транзакцій у банківських API: патерн Saga проти двофазного коміту (2PC). *Американський журнал інженерії та технологій*. 2025. Т. 7, № 06. С. 157–169. DOI: 10.37547/tajet/Volume07Issue06-18.
- [10] Лайгнер Р., Чжоу Ю., Саллес М. А. В., Лю Ю., Каліновські М. Управління даними в мікросервісах: сучасний стан практики, виклики та напрями досліджень. *PVLDB*. 2021. Т. 14, № 13. С. 3348–3361. DOI: 10.14778/3484224.3484232.
- [11] Гірвонен А. Координація між мікросервісами: систематичне картографічне дослідження : магістерська робота. Тампере : Університет Тампере, 2023. URL: <https://trepo.tuni.fi/handle/10024/145653>
- [12] Лунгу С., Ньїренда М. Сучасні тенденції управління розподіленими транзакціями в мікросервісних архітектурах: систематичний огляд літератури. *Відкритий журнал прикладних наук*. 2024. Т. 14. С. 2519–2543. DOI: 10.4236/ojapps.2024.149167.
- [13] Бех А. О. Аналіз сучасних технологій розподілених інформаційних систем у сфері транспортного обслуговування. *ABIA-2025 : матеріали Міжнар. наук.-тех. конф. Київ, 2025. С. 17.16–17.18.*
- [14] Інженерний підрозділ компанії Uber. Оголошення Cadence 1.0: потужної платформи керування робочими процесами, створеної для масштабованості та надійності. 2023. URL: <https://www.uber.com/blog/announcing-cadence/>
- [15] Бех А. О. Огляд сучасних розподілених інформаційних систем. Політ. Сучасні проблеми науки : матеріали Міжнар. наук.-практ. конф. здобувачів вищої освіти і молодих учених. Київ, 2025. С. 181–182.

Грінченко О. О., Бех А. О., Грінченко С. А.

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОТОКОЛІВ УПРАВЛІННЯ ТРАНЗАКЦІЯМИ PIC (ОГЛЯД)

У дослідженні проведено порівняльний аналіз протоколів управління розподіленими транзакціями – двофазного коміту (2PC), трифазного коміту (3PC), патерну Saga та протоколу Try-Confirm-Cancel (TCC) – у контексті транспортних розподілених інформаційних систем (PIC), для яких критичними є надійність, низька латентність та масштабованість. Транспортні PIC характеризуються різноманітністю сервісів (планування, тарифікація, оплати, бронювання, телеметрія) й необхідністю використання міжсервісних транзакцій. На основі сучасних публікацій і практичного використання сформовано критерії порівняння (гарантії узгодженості та ізоляції, затримка, пропускна здатність, блокування при збої, складність реалізації) та представлено узагальнену таблицю порівняння характеристик основних протоколів. Показано, що 2PC забезпечує суворі ACID-гарантії та звичну модель програмування, однак може призводити до блокування системи у випадку збоїв координатора. 3PC зменшує проблему блокування, вводячи додаткову фазу, проте це ускладнює реалізацію і збільшує комунікаційні витрати. Натомість шаблон Saga забезпечує узгодженість даних шляхом послідовності локальних транзакцій із компенсаціями, підвищуючи відмовостійкість і продуктивність системи за рахунок допущення часової неузгодженості даних. TCC вводить попереднє резервування ресурсів з підтвердженням/скасуванням, підвищуючи узгодженість порівняно з Saga, однак потребує ще складнішої реалізації та все це працює в режимі остаточної узгодженості. Сформульовано рекомендації для інженерів щодо вибору оптимального механізму транзакцій з огляду на пріоритети системи: коли першим є миттєва узгодженість – доцільний локалізований 2PC або 3PC; коли важливі доступність і швидкодія – домінують Saga/TCC; гібридне поєднання забезпечує збалансований компроміс між продуктивністю, масштабованістю та вимогами до цілісності даних у PIC.

Ключові слова: 2PC; 3PC; Saga; TCC; мікросервіси; транспортні інформаційні системи; розподілені транзакції; ACID/BASE; узгодженість даних.

Grinenko O., Bekh A., Grinenko S.**COMPARATIVE ANALYSIS OF TRANSACTION MANAGEMENT PROTOCOLS
IN DIS (REVIEW)**

The study presents a comparative analysis of distributed transaction management protocols - the Two-Phase Commit (2PC), Three-Phase Commit (3PC), Saga pattern, and Try-Confirm-Cancel (TCC) - in the context of transport distributed information systems (DIS), where reliability, low latency, and scalability are critical. Transportation DIS are characterized by a diversity of services (scheduling, pricing, payments, reservations, telemetry) and the need for inter-service transactions. Based on modern publications and practical applications, comparison criteria were established (consistency and isolation guarantees, latency, throughput, failure blocking, implementation complexity), and a generalized comparison table of the main protocols' characteristics was presented. It is shown that 2PC provides strict ACID guarantees and a familiar programming model, but may lead to system blocking in case of coordinator failures. 3PC mitigates the blocking problem by introducing an additional phase, but this complicates implementation and increases communication overhead. In contrast, the Saga pattern ensures data consistency through a sequence of local transactions with compensations, improving fault tolerance and performance at the expense of allowing temporary data inconsistency. TCC introduces preliminary resource reservation with confirmation/cancellation, increasing consistency compared to Saga, but requires even more complex implementation and still operates under eventual consistency. Recommendations are formulated for engineers on selecting the optimal transaction mechanism depending on system priorities: when immediate consistency is paramount - a localized 2PC or 3PC is advisable; when availability and speed are more important - Saga/TCC dominate; a hybrid combination ensures a balanced compromise between performance, scalability, and data integrity requirements in DIS.

Keywords: 2PC; 3PC; Saga; TCC; microservices; transport information systems; distributed transactions; ACID/BASE; data consistency.

Дата першого надходження: 10.07.2026 р.

Дата прийняття до друку: 13.08.2026 р.

Дата публікації: 10.09.2026 р.