

5. Косвенное взаимодействие

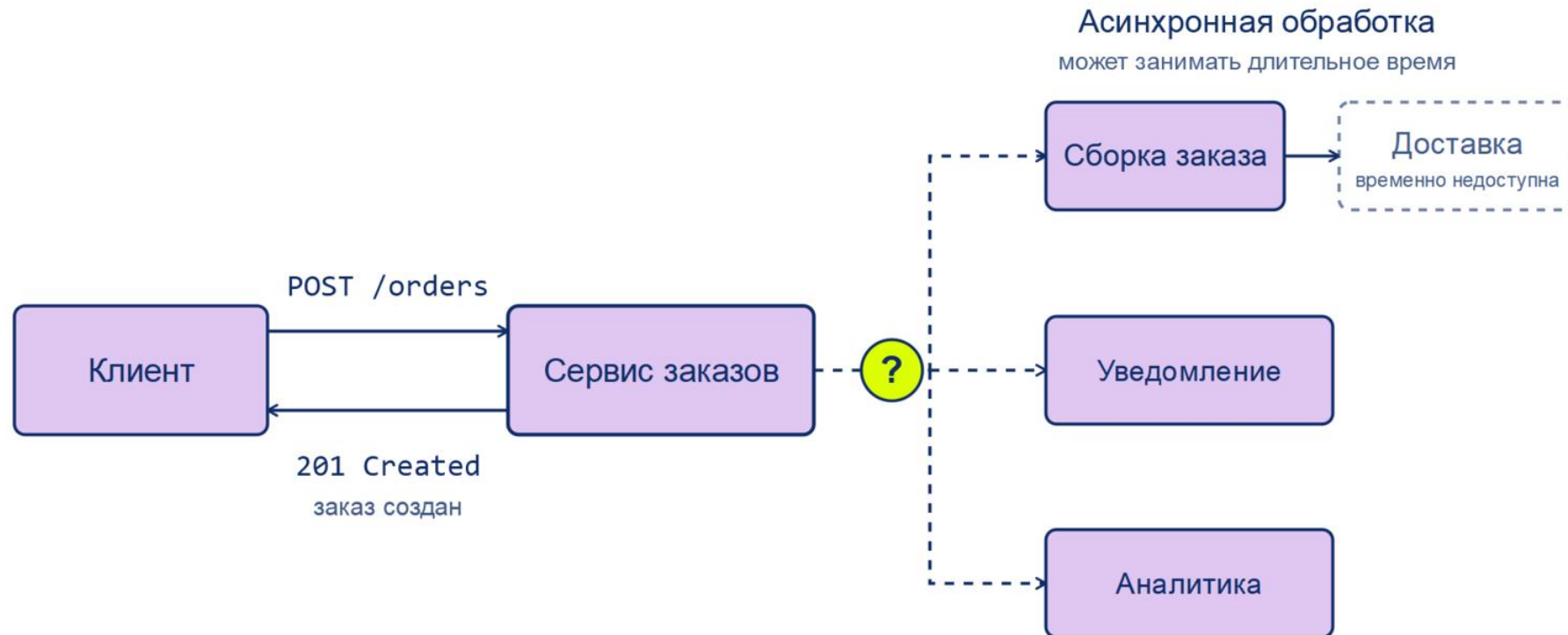
Сухорослов Олег Викторович

05.10.2026

Обработка заказа после ответа клиенту

Сборка и доставка занимают время. Клиент получает подтверждение создания заказа, а дальнейшая обработка продолжается асинхронно.

Как передать работу другим сервисам, в том числе временно недоступным?



Косвенное взаимодействие

Процессы обмениваются данными через общую абстракцию (очередь, канал, тема, журнал, разделяемое состояние...)

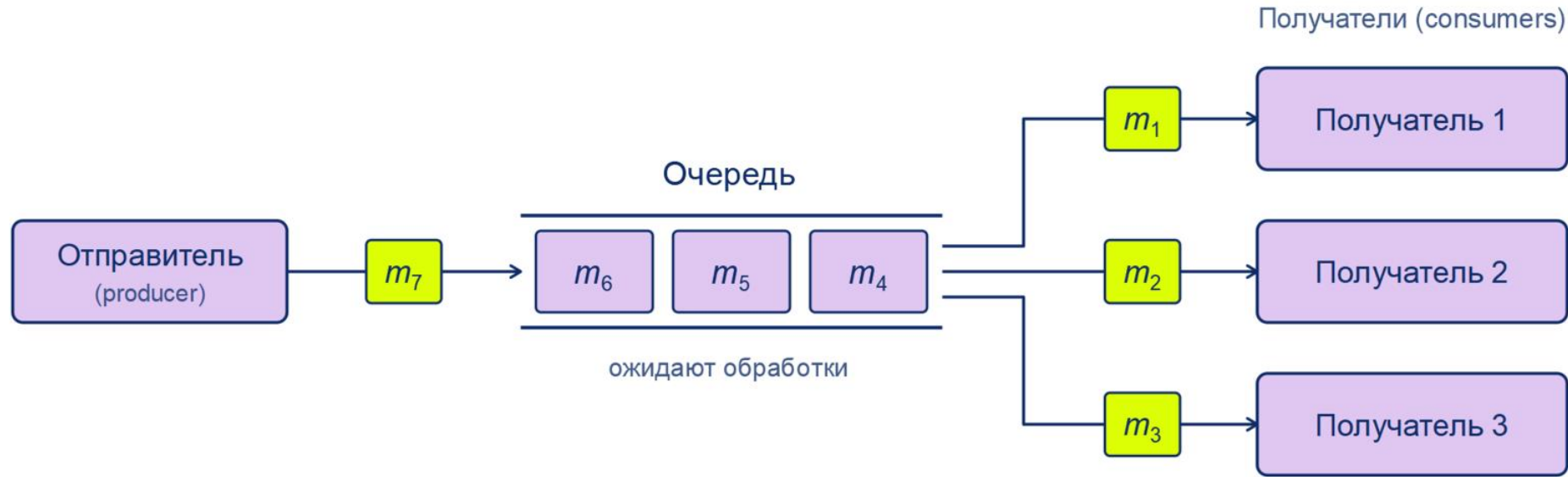
- **Развязывание по пространству:** не требуется знать конкретного получателя.
- **Развязывание по времени:** участники могут работать в разное время.



План лекции

- **Обмен сообщениями:** очередь, издатель-подписчик, журнал.
- **Гарантии:** хранение, доставка, обработка и порядок сообщений.
- **Разделяемое состояние:** распределённая общая память и пространство кортежей.
- **Сравнение моделей и выбор подхода.**

Очередь сообщений



Очередь накапливает сообщения для последующей обработки.

Брокер сообщений – сервис, который управляет очередями, принимает сообщения от отправителей и передаёт их получателям.

- Сообщения из одной очереди распределяются между получателями.
- Несколько получателей могут выполнять работу параллельно.

Буферизация и нагрузка

Всплеск нагрузки



После всплеска



- При всплеске нагрузки сообщения накапливаются в очереди.
- Когда скорость поступления становится ниже скорости обработки, очередь сокращается.
- При постоянной перегрузке растут длина очереди и время ожидания.

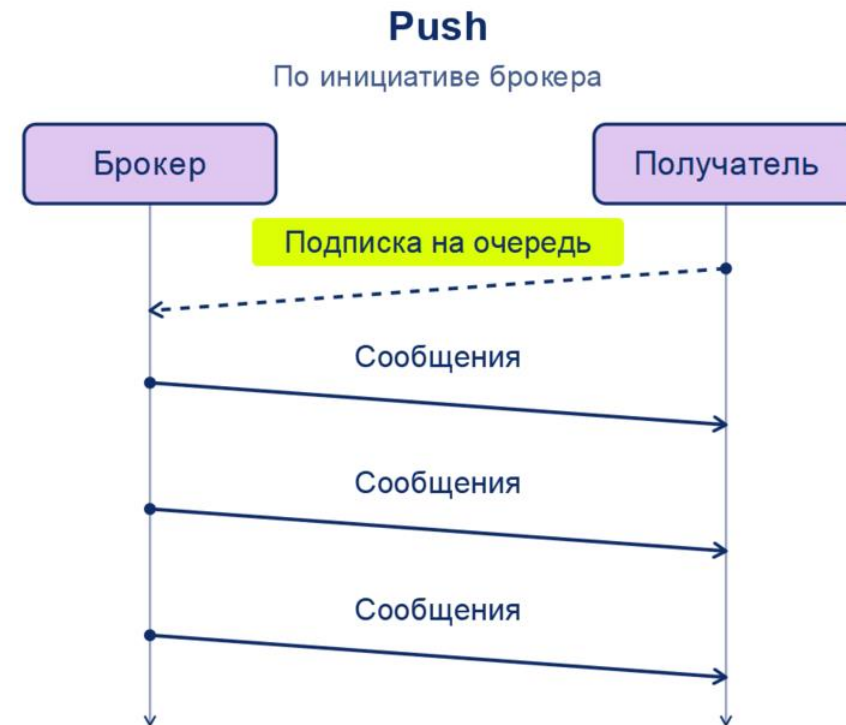
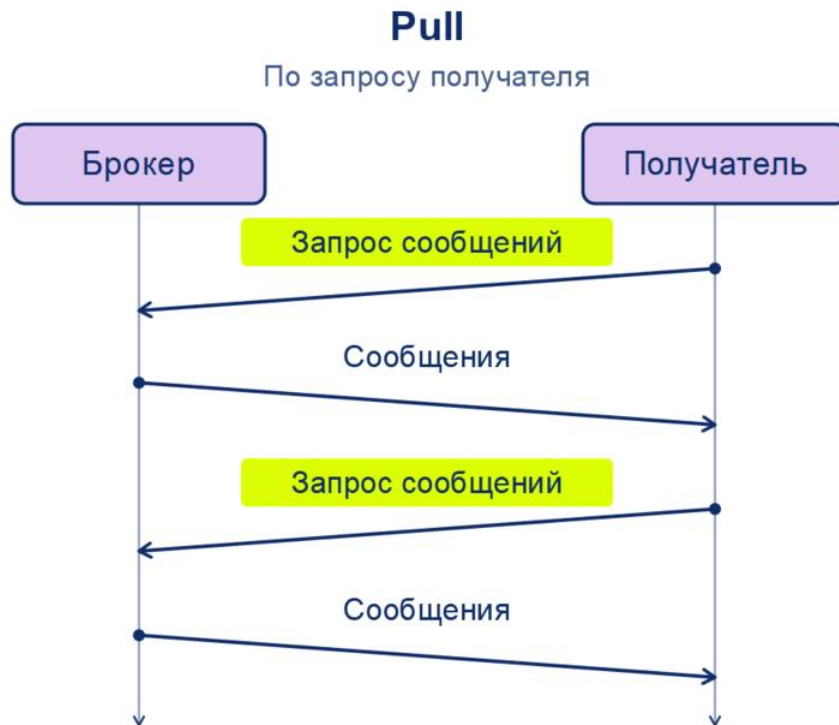
Получение сообщений из очереди

Pull – по запросу получателя

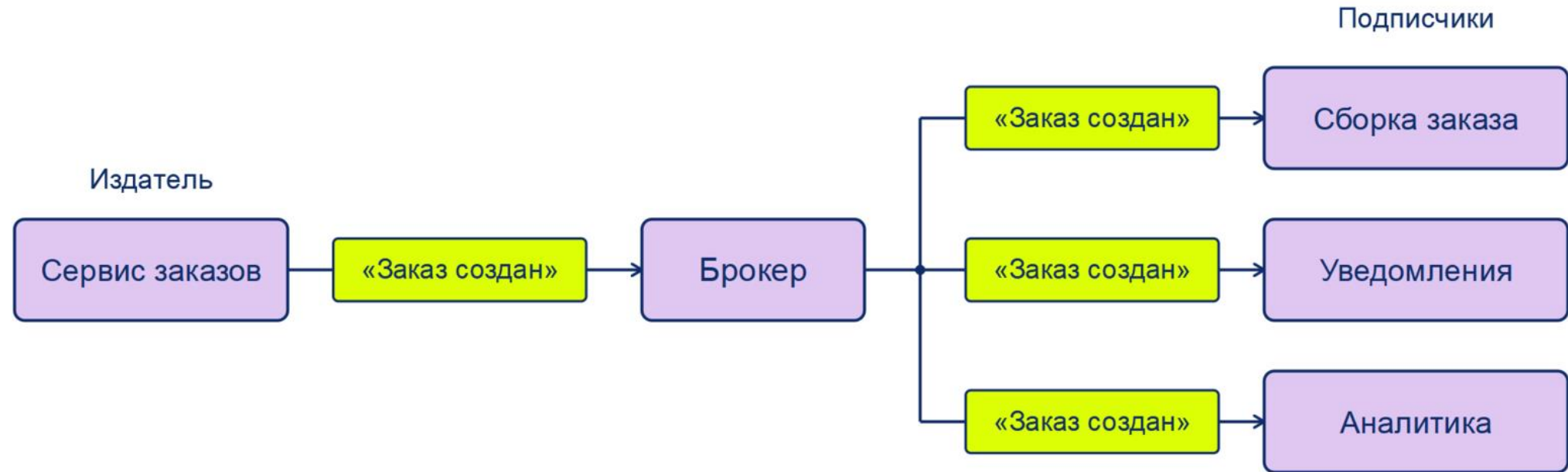
Получатель запрашивает сообщения, когда готов их обрабатывать.

Push – по инициативе брокера

Получатель подписывается на очередь, а брокер передаёт ему сообщения по соединению.

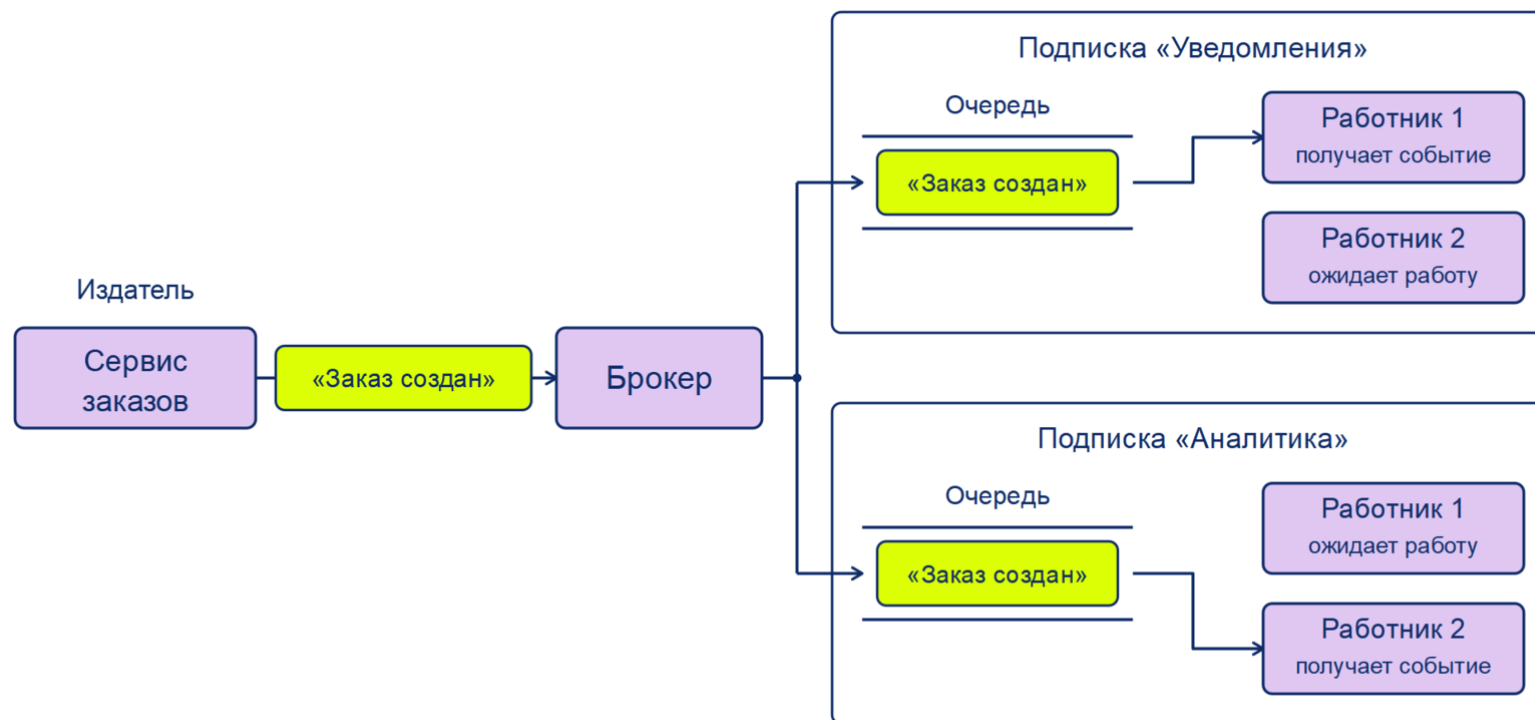


Издатель–подписчик (publish–subscribe)



- **Издатель** публикует сообщения о событиях.
- **Подписчики** заранее указывают, какие сообщения им нужны.
- Каждое сообщение предназначено всем заинтересованным подписчикам.

Подписки и экземпляры сервиса



Событие получает каждый сервис, внутри сервиса — один работник

- Каждая независимая подписка получает свою копию события.
- Одну подписку могут обслуживать несколько экземпляров сервиса.
- Работа внутри подписки распределяется между экземплярами.

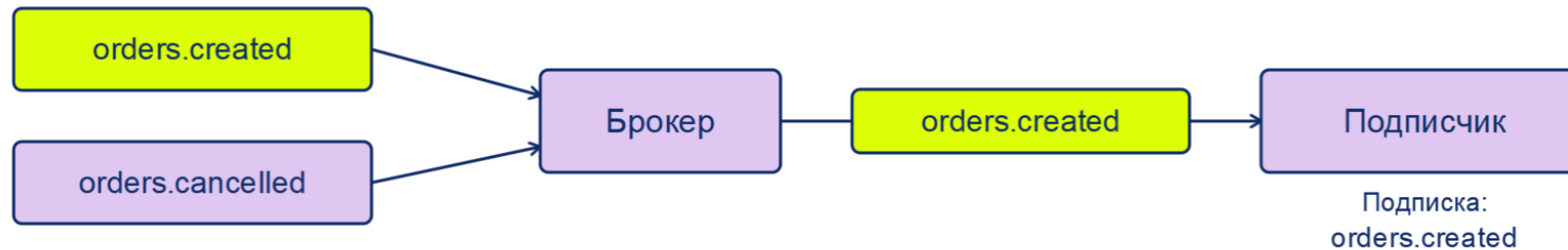
Pub/sub между сервисами, очередь задач внутри сервиса.

Подписки и отбор сообщений

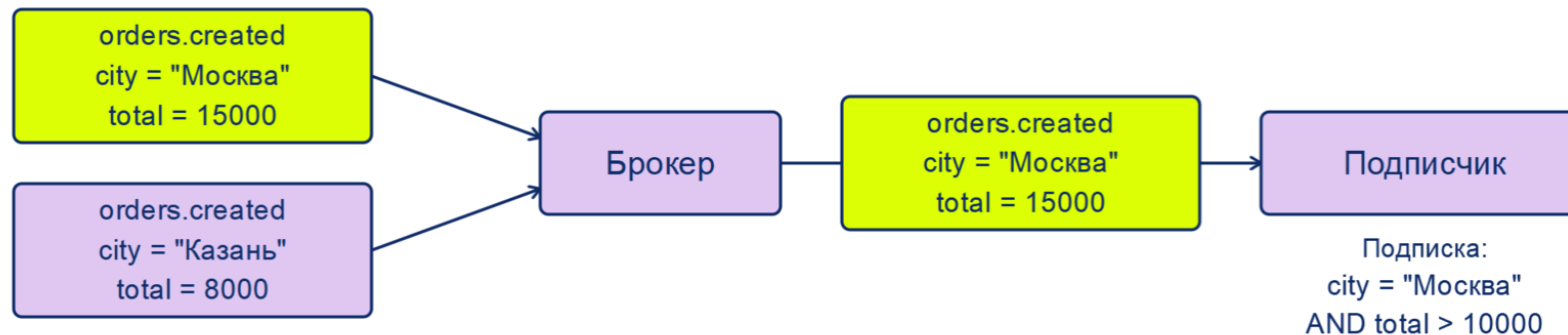
Подписка определяет, какие сообщения интересуют получателя.

- **По теме (topic)** – выбор именованных категорий сообщений.
- **По содержимому** – условие на значения полей сообщения.

По теме



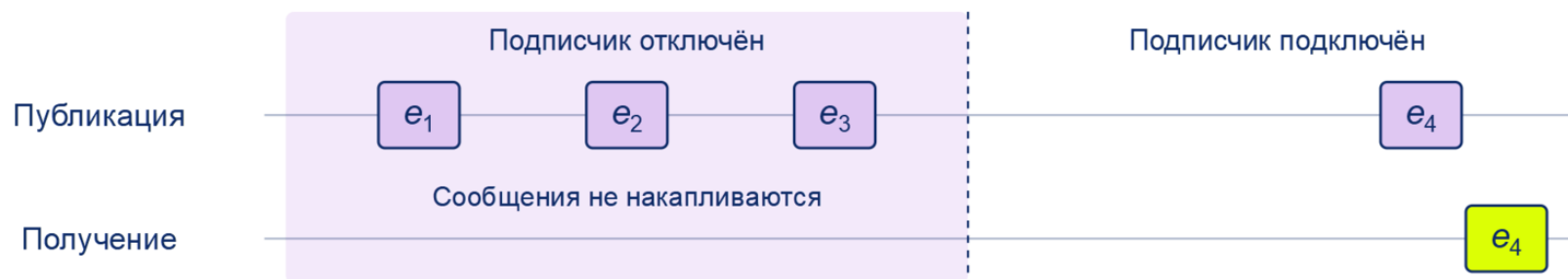
По содержимому



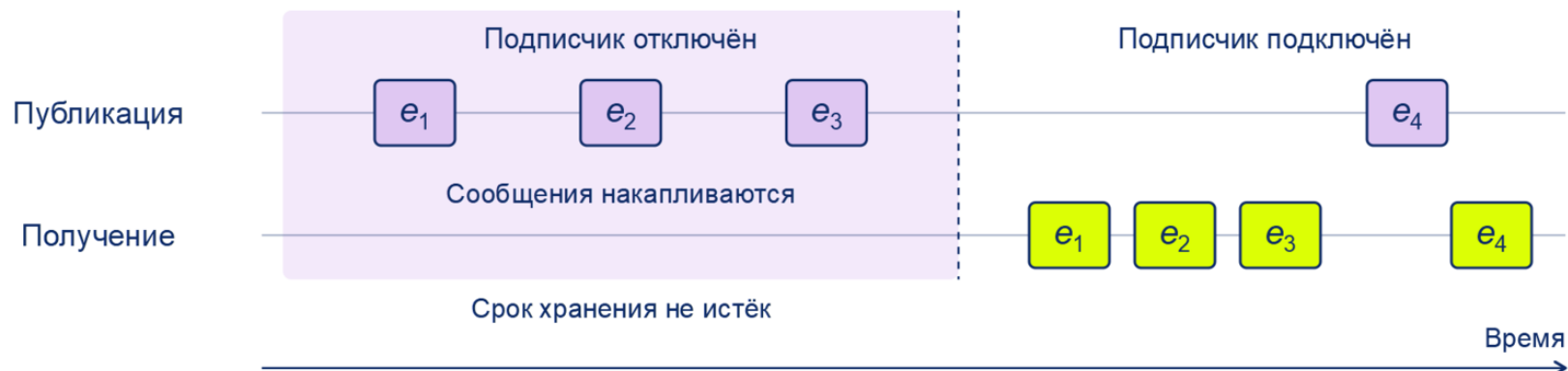
Подписчик временно отключен

Получит ли подписчик события, опубликованные во время его отсутствия?

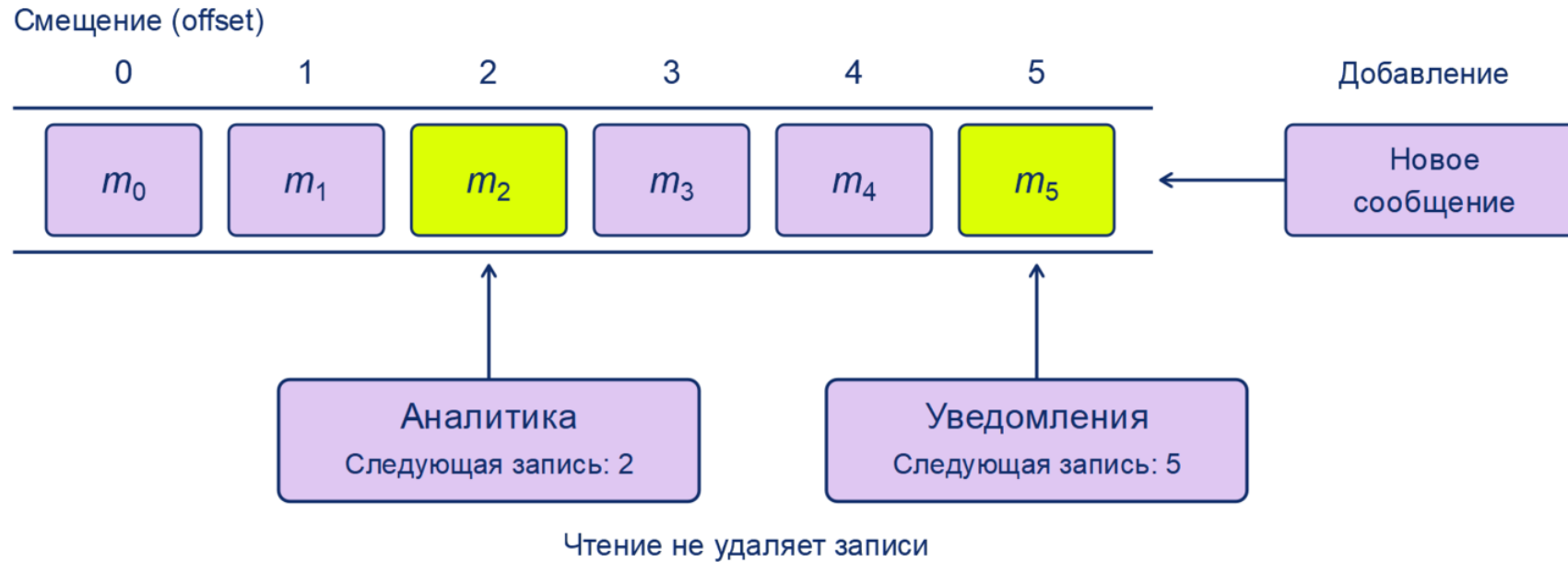
Без накопления



С накоплением



Журнал сообщений

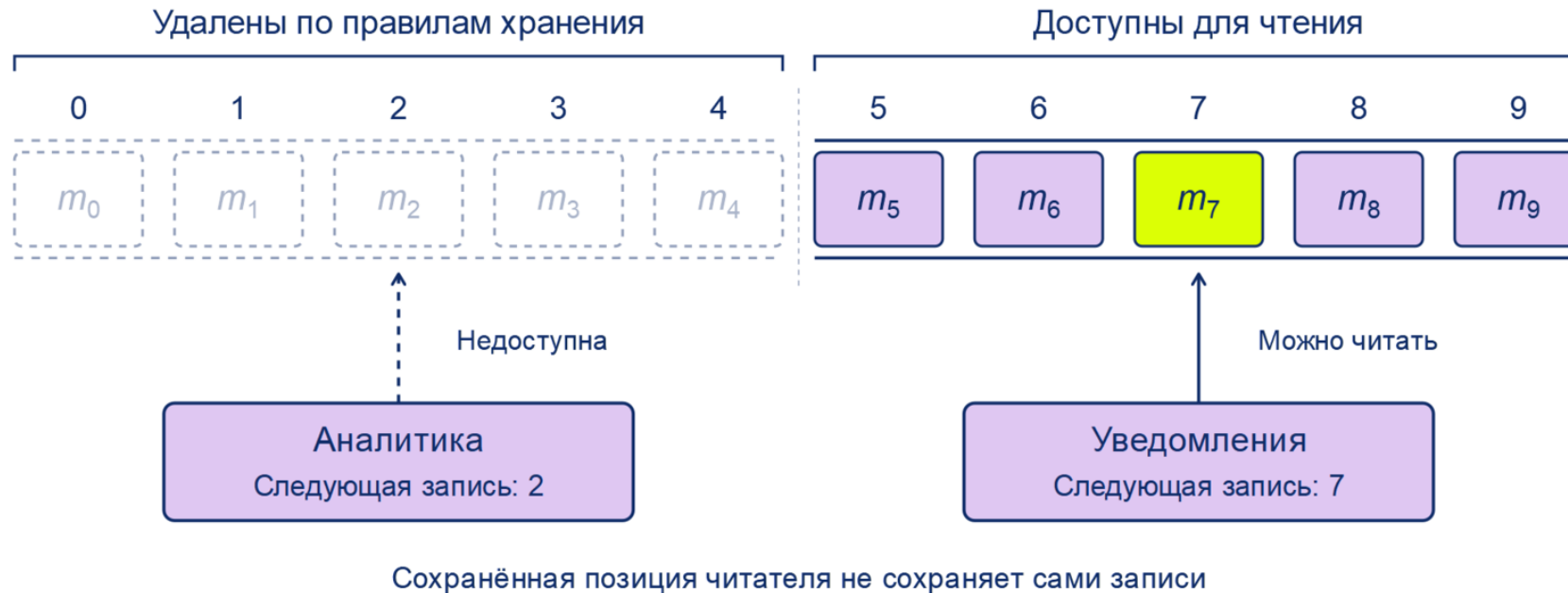


Журнал – сохраняемая последовательность сообщений.

- Новые сообщения добавляются в конец журнала.
- Каждый читатель независимо отслеживает свою позицию – **смещение (offset)**.
- Чтение не удаляет сообщения: их можно прочитать повторно.

История доступна в пределах срока и/или объёма хранения.

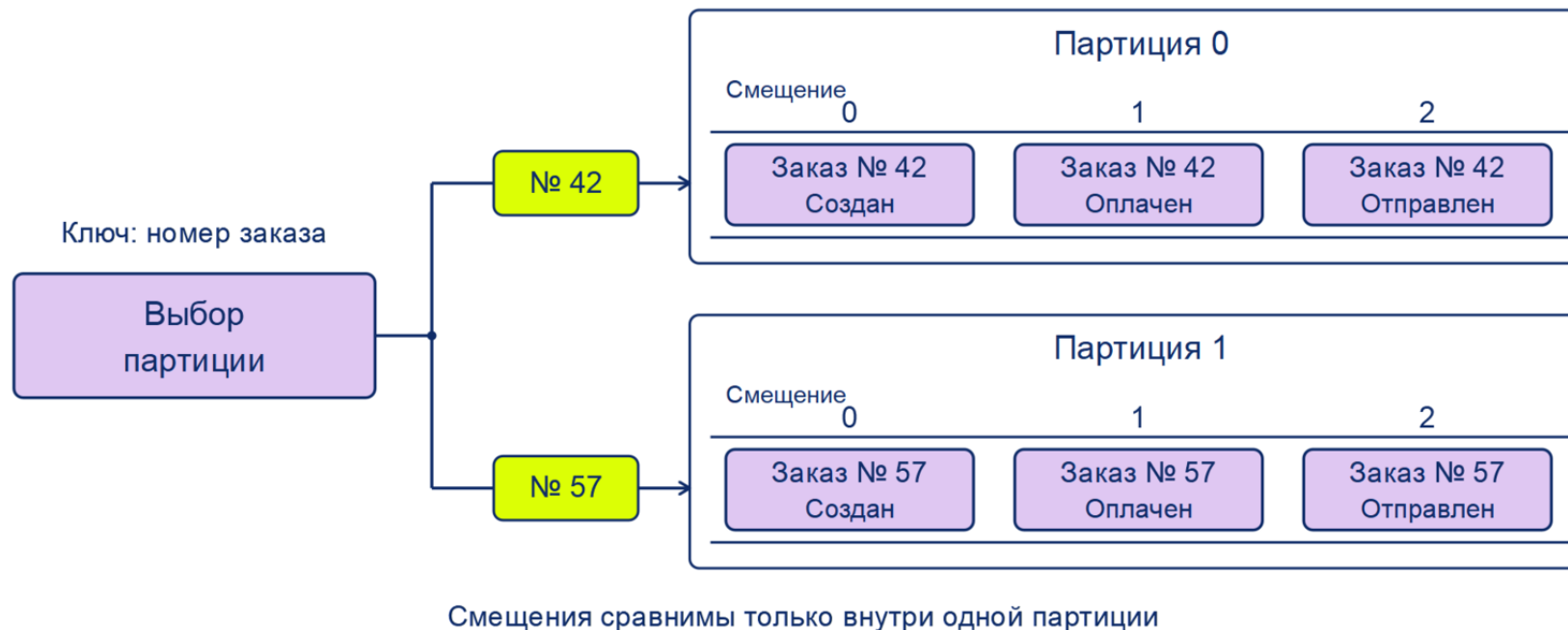
Повторное чтение и срок хранения



- Читатели независимо продвигаются по журналу.
- Позицию можно вернуть назад и повторить чтение.
- Повторное чтение возможно, **пока записи хранятся**.

При хранении по времени/объёму старые записи могут удаляться, даже если читатель их ещё не прочитал.

Партиции журнала

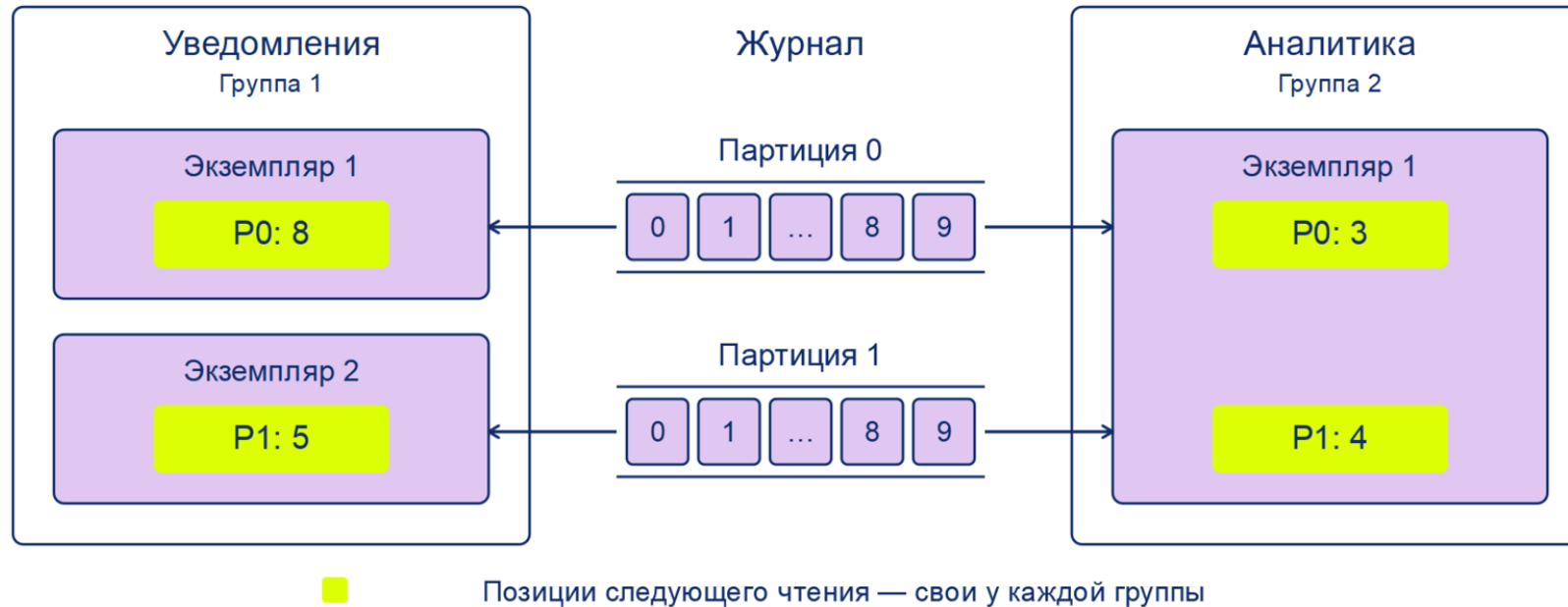


Партиция – отдельная упорядоченная последовательность записей внутри журнала.

- У каждой партиции свои смещения.
- Разные партиции можно хранить и читать параллельно.
- Ключ сообщения позволяет направлять связанные записи в одну партицию.

Порядок внутри партиции не задаёт общего порядка между партициями.

Группы читателей



Несколько экземпляров одного сервиса объединяются в **группу читателей**.

- Внутри группы партии распределяются между экземплярами.
- Каждую партицию читает один экземпляр группы.
- У разных групп – независимые позиции чтения.

Число партиций ограничивает параллелизм чтения внутри группы.

Очередь, pub/sub и журнал

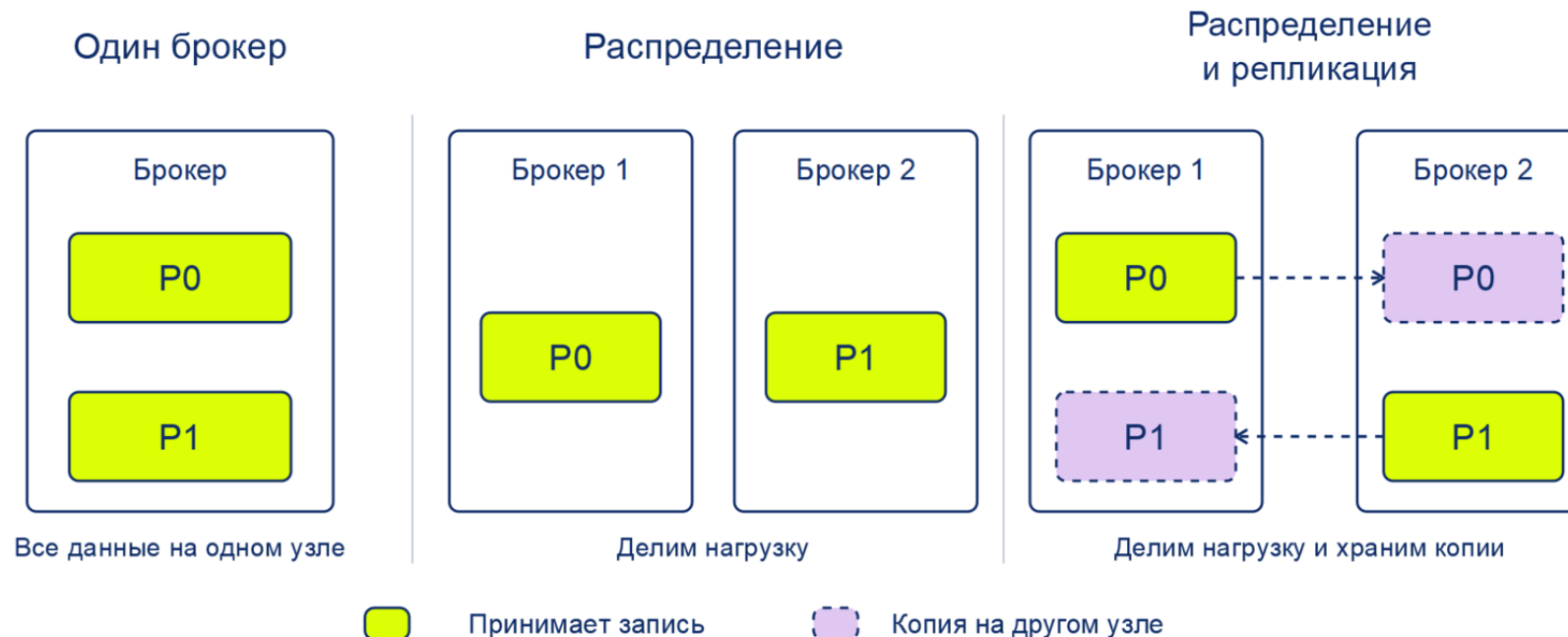
Модель	Основная идея	Пример
Очередь задач	Распределять работу между исполнителями	Поручить сборку заказа одному из работников
Pub/sub	Передавать событие всем заинтересованным подписчикам	Сообщить о заказе сборке, уведомлениям и аналитике
Журнал	Хранить историю для независимого и повторного чтения	Пересчитать аналитику по сохранённым событиям

Модели можно сочетать: подписчик может иметь свою очередь или читать из журнала.

От одного брокера к кластеру

Посредник может быть распределённой системой.

- **Один брокер:** ограниченные ресурсы, единая точка отказа.
- **Распределение данных:** очереди или партии размещаются на разных узлах.
- **Репликация:** копии данных хранятся на нескольких узлах.



Отправка, доставка и обработка



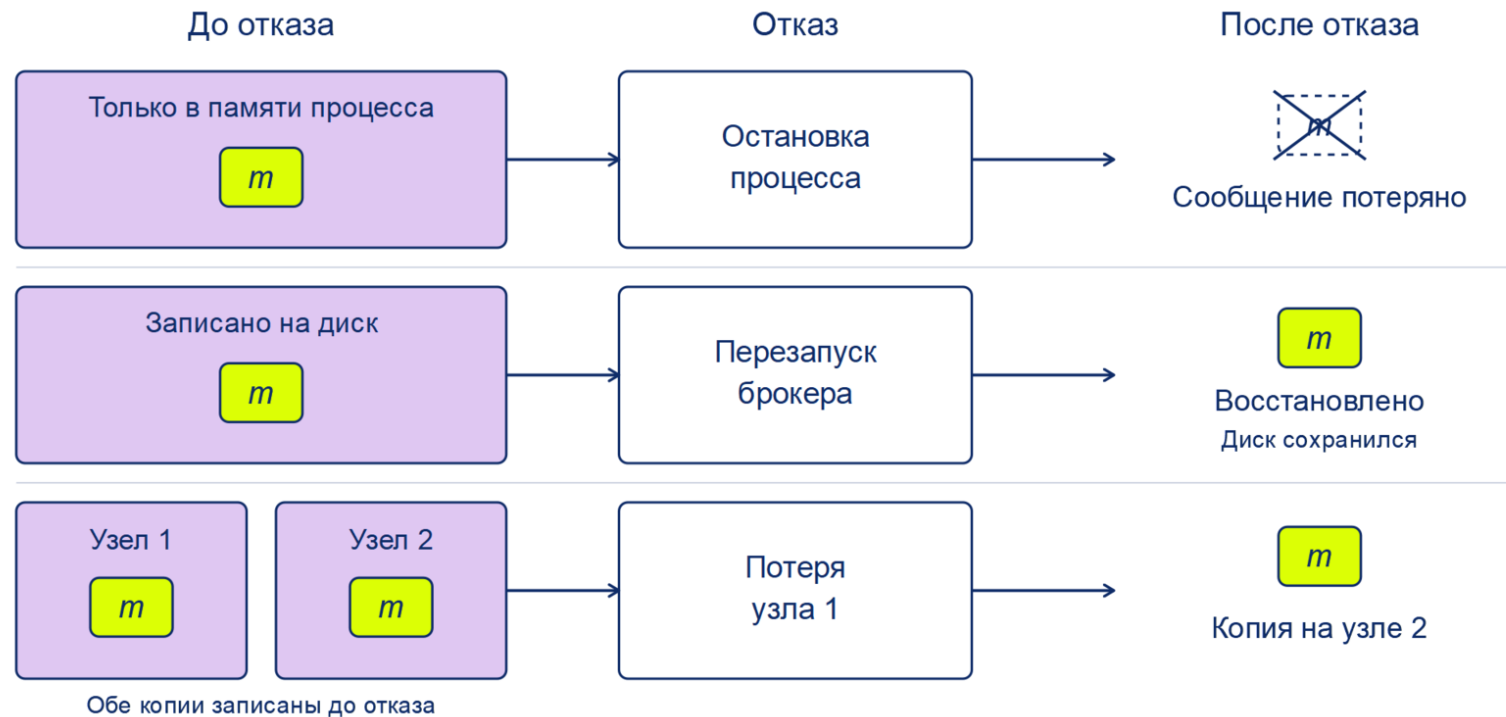
Между этими этапами возможны задержки и отказы.

Подтверждение приёма брокером ещё не означает, что получатель обработал сообщение.

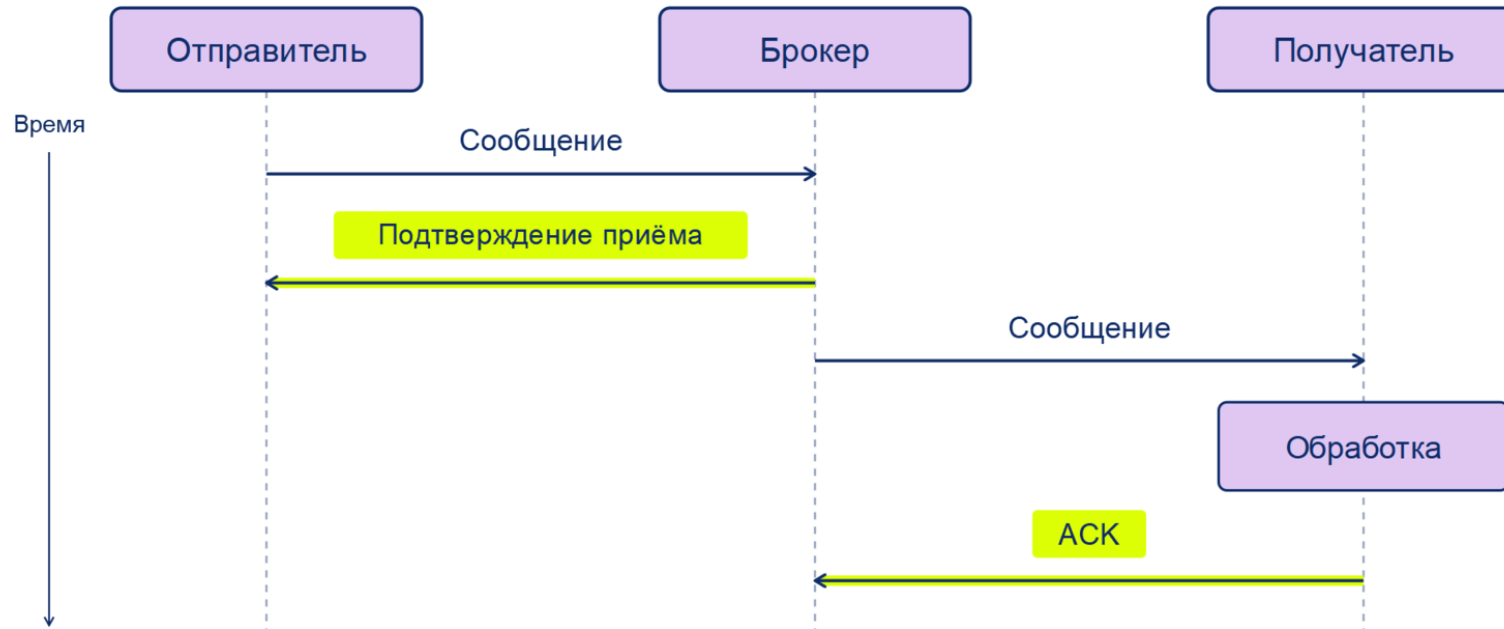
Какие отказы переживёт сообщение?

- **Только в памяти:** может потеряться при остановке процесса.
- **На диске:** переживёт перезапуск, если диск сохранился.
- **На нескольких узлах:** может пережить потерю узла, если сохранилась другая копия.

Что успело сохраниться к моменту подтверждения отправителю?



Два вида подтверждений



Подтверждение брокера → отправителю

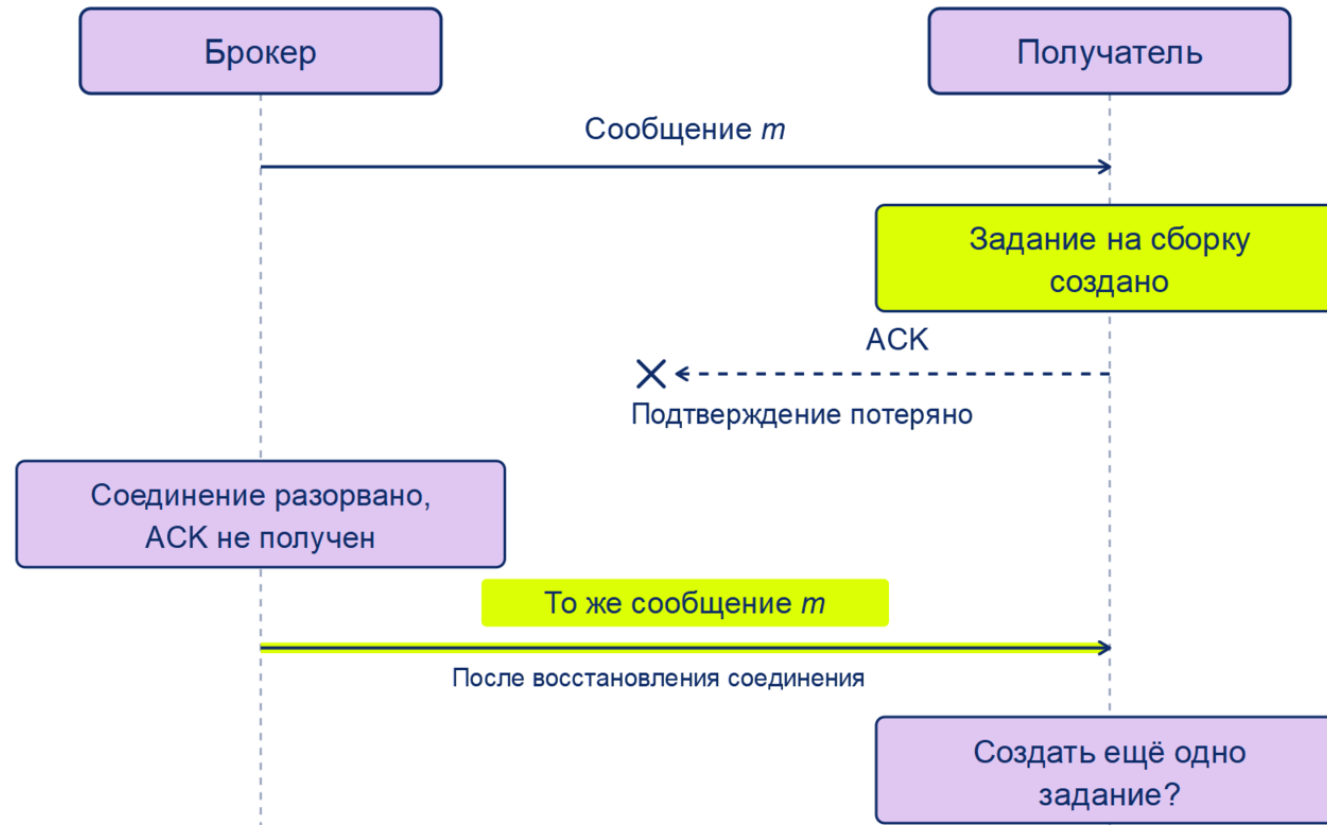
Брокер принял сообщение согласно своим гарантиям.

Подтверждение получателя → брокеру (ACK)

Получатель сообщает, что доставку можно считать завершённой.

Если ACK отправлен до завершения обработки, при отказе работа может остаться невыполненной.

Повторная доставка



Действие выполнено, но ACK потерян – брокер может доставить сообщение повторно.
Повторная доставка может привести к повторному действию.

Гарантии доставки

Гарантия	Сколько раз доставляется сообщение
At-most-once – не более одного раза	Ноль или один: возможна потеря
At-least-once – хотя бы один раз	Один или несколько: возможны повторы
Exactly-once – ровно один раз	Один: без потери и повторной доставки

Ровно одна доставка и однократный результат обработки – разные гарантии.

Повторные попытки возможны и при однократном результате.

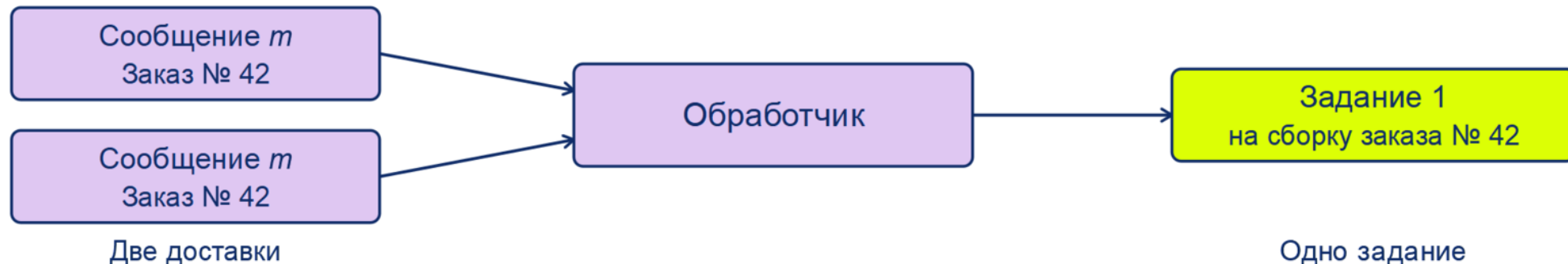
Идемпотентная обработка

Повторная обработка одного сообщения не создаёт дополнительного эффекта.

Без учёта повторов



Идемпотентная обработка



Почему недостаточно запомнить ID сообщения

Между действием и сохранением ID возможен сбой:

- **Сначала действие:** при повторной доставке оно выполнится ещё раз.
- **Сначала ID:** при повторной доставке невыполненное действие будет пропущено.

Результат и отметку об обработке нужно согласовать.

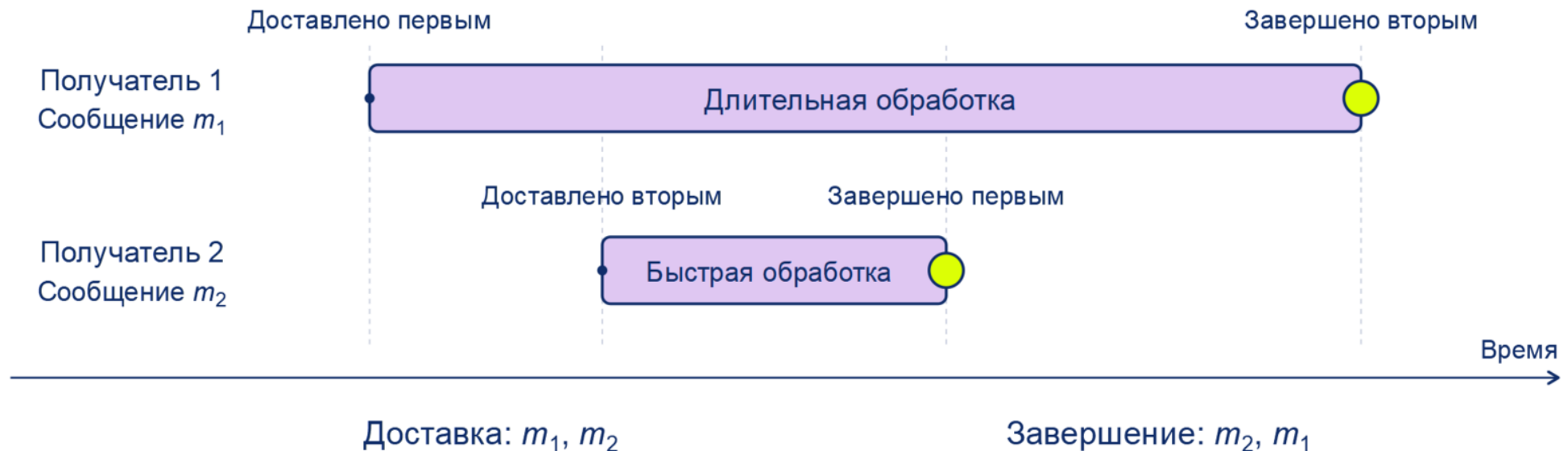


Порядок доставки и обработки

Порядок доставки не гарантирует порядок завершения обработки.

- Параллельные получатели могут завершать работу в разном порядке.
- Повторная доставка может вернуть старое сообщение после новых.

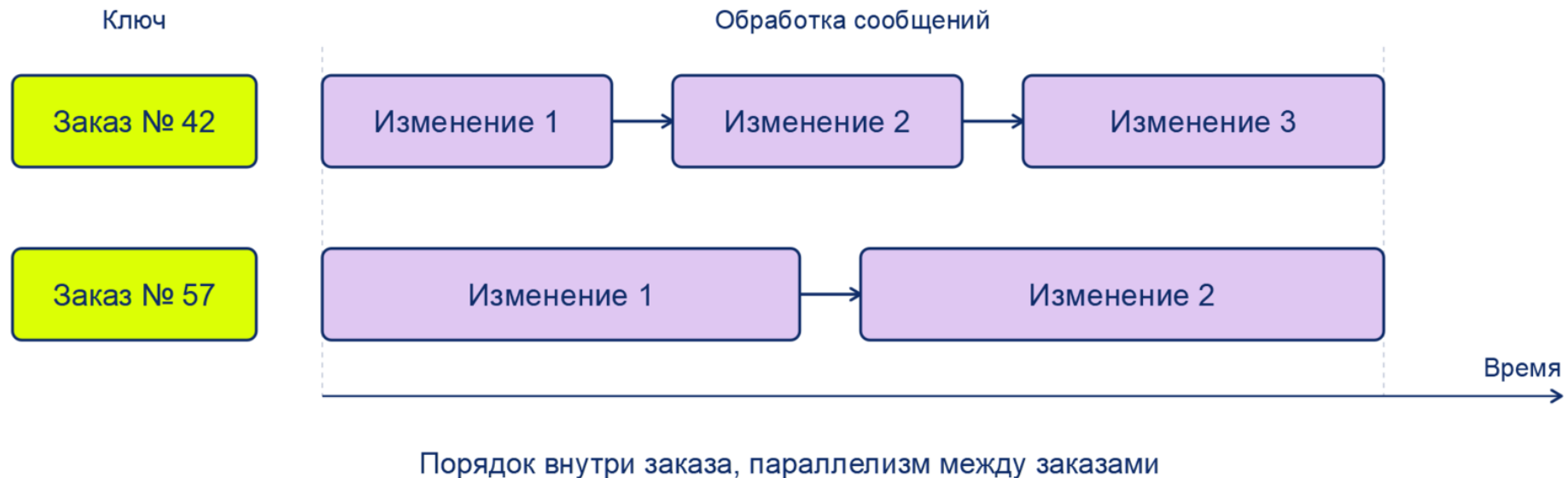
Уточняйте границы гарантии (очередь, партиция, сообщения одного заказа...)



Порядок и параллелизм

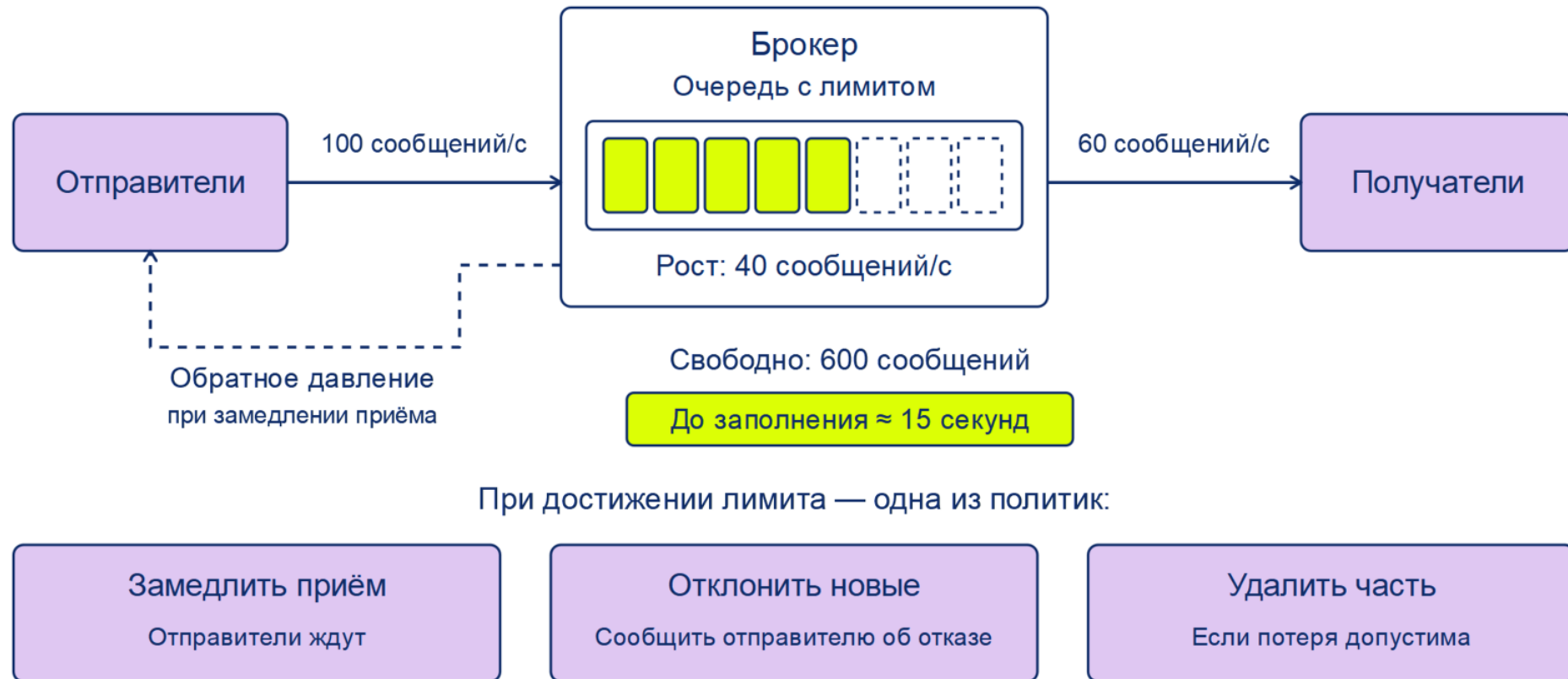
Порядок часто нужен только для сообщений об одном объекте.

- Связанные сообщения объединяют по **ключу**, например номеру заказа.
- Сообщения одного ключа обрабатывают последовательно.
- Сообщения разных ключей можно обрабатывать параллельно.

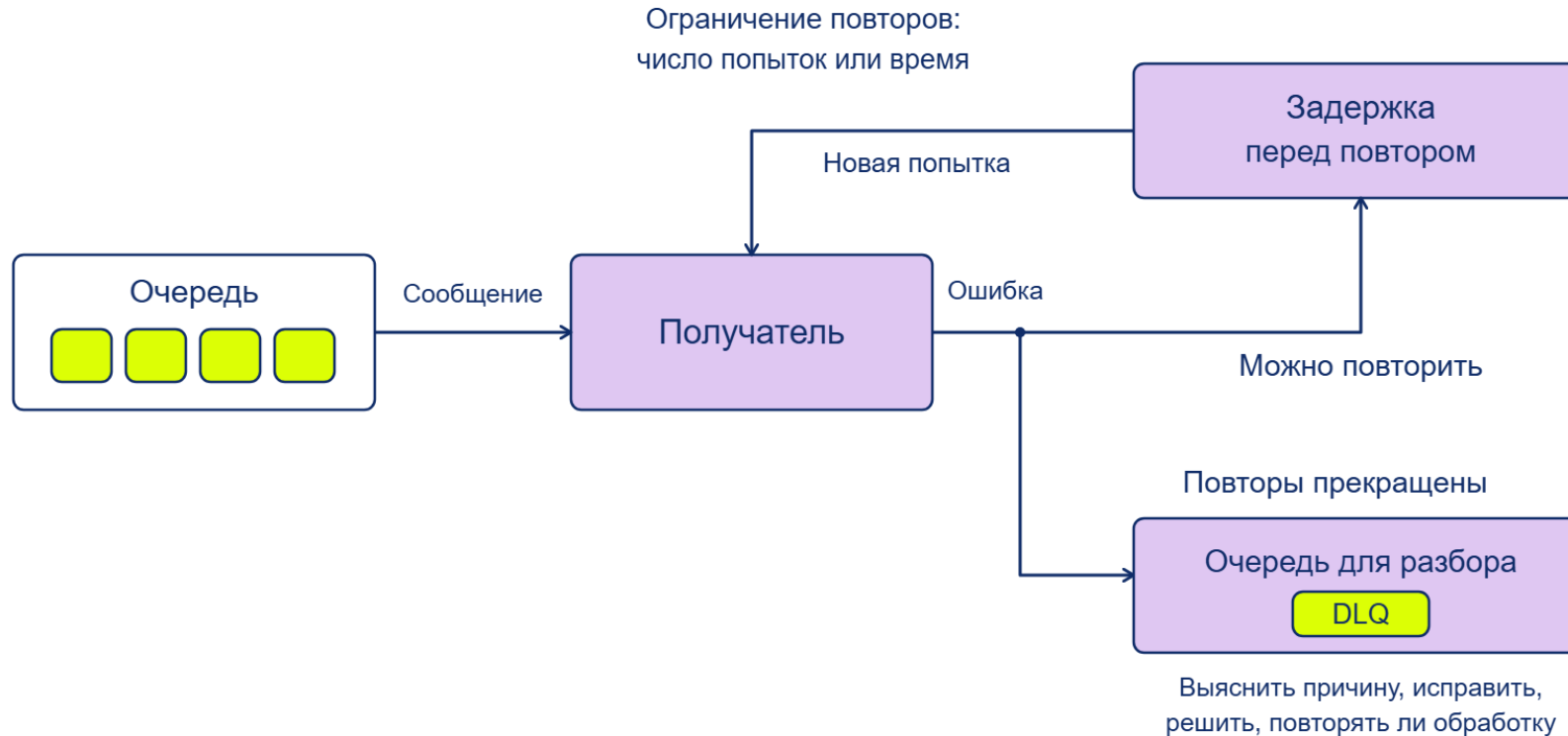


Когда очередь заполняется

Если сообщения поступают быстрее, чем обрабатываются, очередь растёт и может достигнуть лимита размера. **Конечный буфер не решает проблему постоянной перегрузки.**



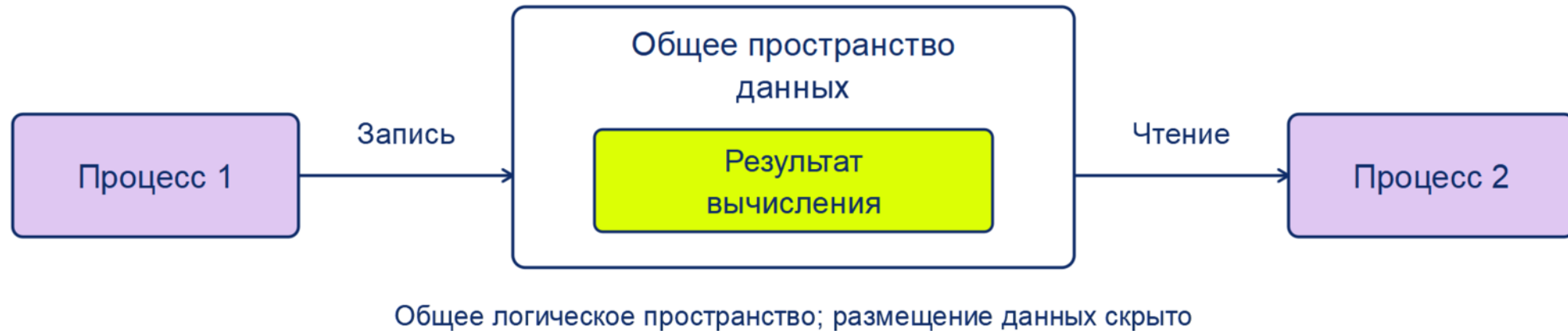
Обработка постоянно завершается ошибкой



- **Временный сбой:** повторить позже, с задержкой.
- **Неустраняемая ошибка или исчерпан лимит попыток:** прекратить автоматические повторы.
- **Проблемное сообщение:** сохранить отдельно для разбора – *dead-letter queue* (DLQ).

Повторная доставка не гарантирует успешную обработку.

Разделяемое состояние

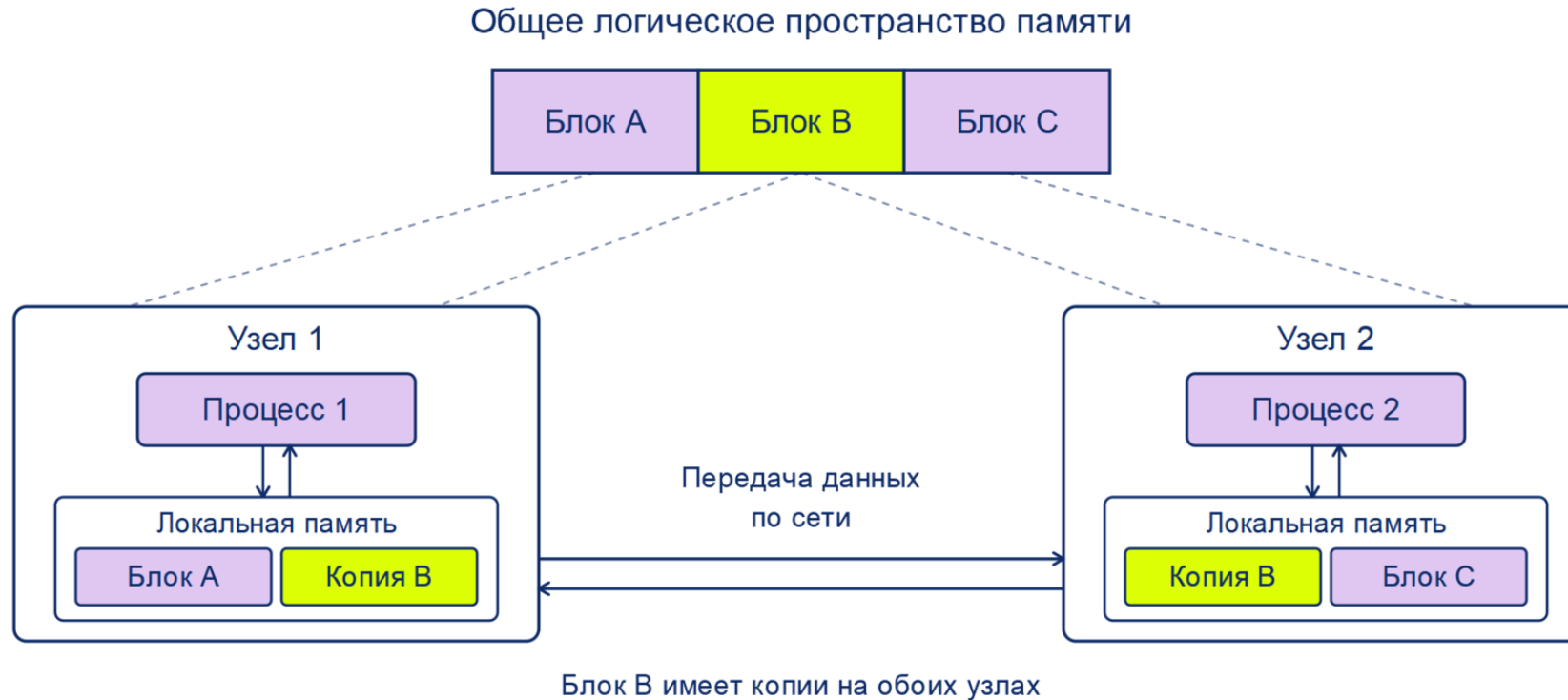


Процессы читают и изменяют общее логическое пространство данных.

- Один процесс записывает результат, другой – читает его.
- Данные могут физически находиться на разных узлах.

Когда и в каком порядке изменения становятся видны другим процессам?

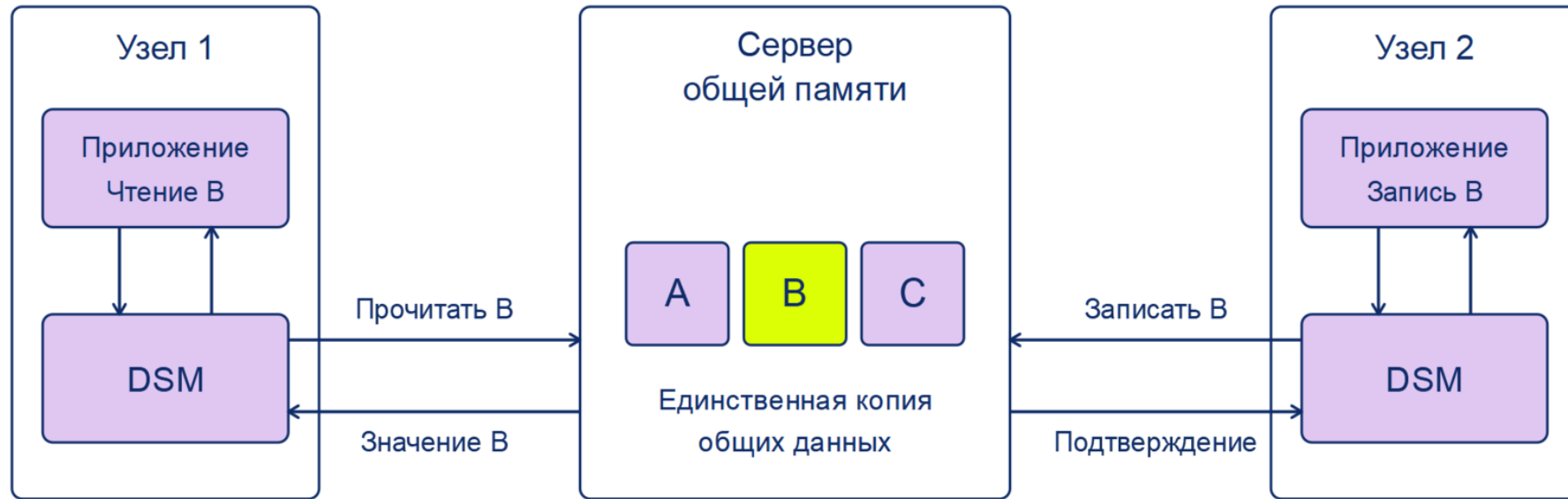
Распределённая разделяемая память



DSM (Distributed Shared Memory) предоставляет процессам на разных узлах абстракцию общей памяти.

- Процессы читают и изменяют общие данные.
- Система скрывает их размещение и передачу между узлами.
- Обращение к памяти может потребовать обмена по сети.

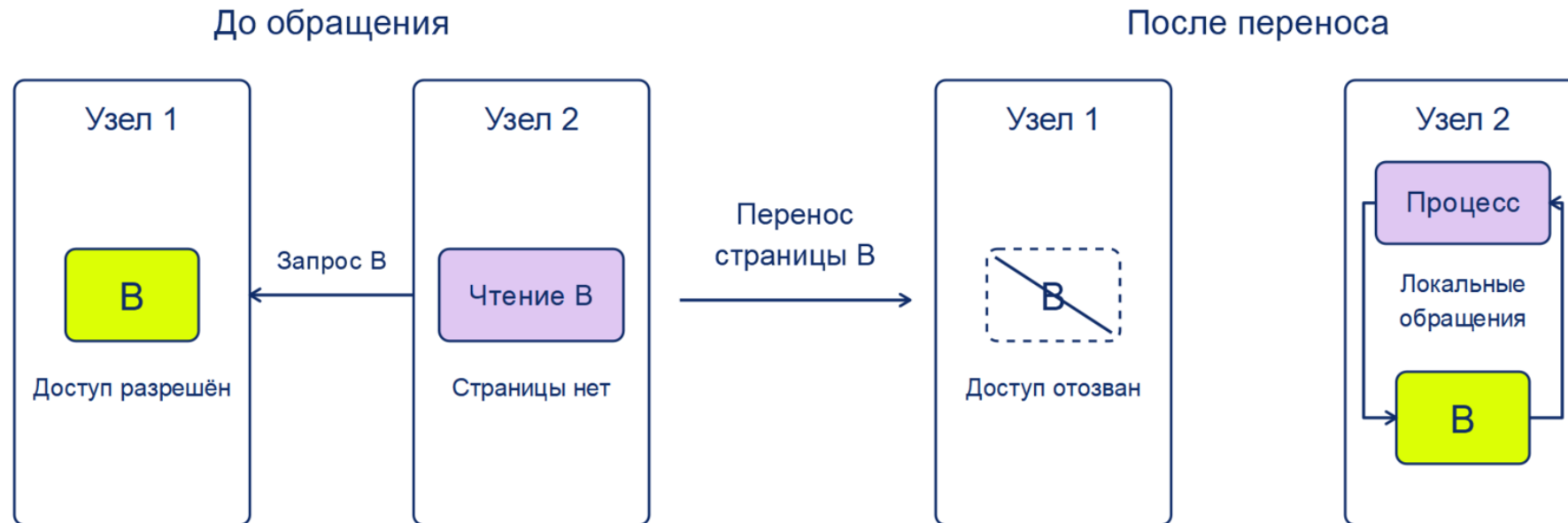
DSM: центральный сервер



- Единственная копия общих данных находится на сервере.
- Чтение и запись выполняются через запросы к серверу.
- Система DSM скрывает сетевой обмен от приложения.

**Каждое удалённое обращение требует обмена по сети.
Сервер ограничивает производительность и доступность.**

DSM: миграция страниц



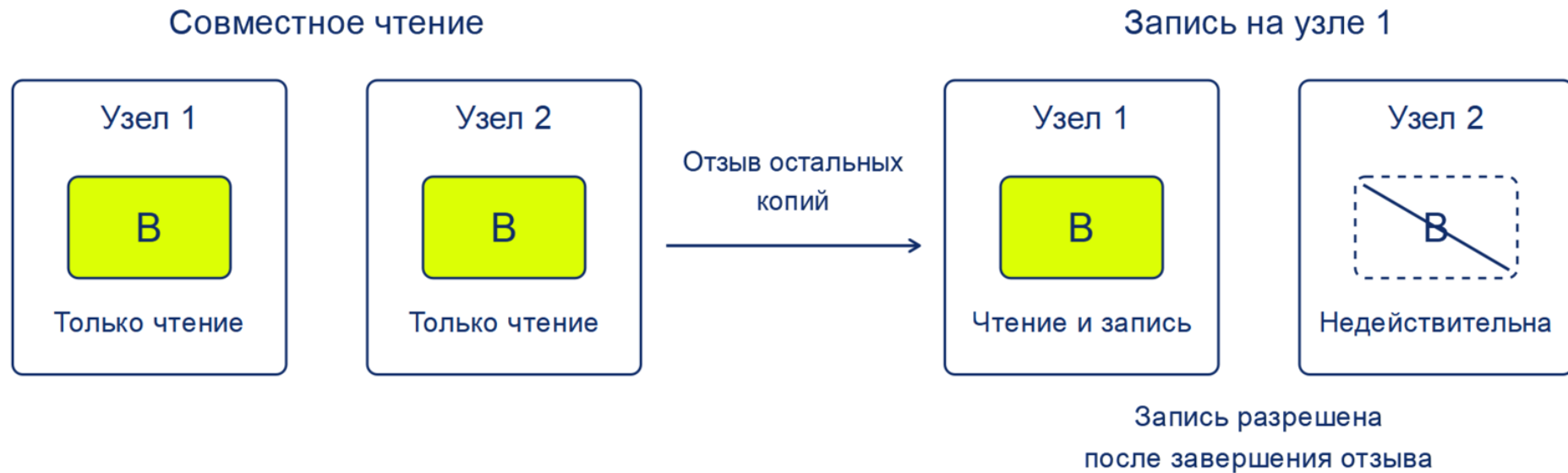
Страница – блок памяти фиксированного размера. DSM передаёт страницы целиком.

Страница доступна одному узлу.

- При обращении другого узла переносится к нему.
- Повторные обращения выполняются локально.

Чередование обращений разных узлов вызывает пересылки туда и обратно.

DSM: репликация для чтения

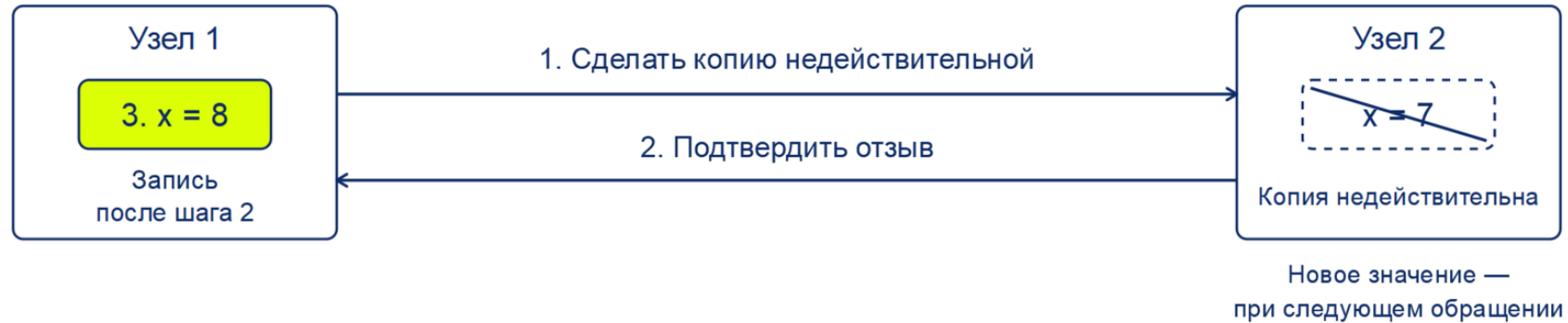


- Несколько узлов читают локальные копии страницы.
- Перед записью остальные копии делаются недействительными.
- Записывающий узел получает исключительный доступ.

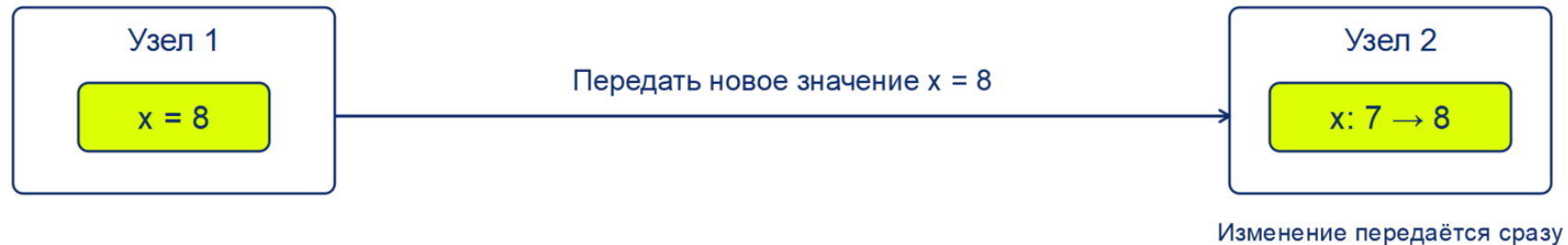
Чтение – локально, запись – с согласованием.

Инвалидация или обновление копий

Инвалидация



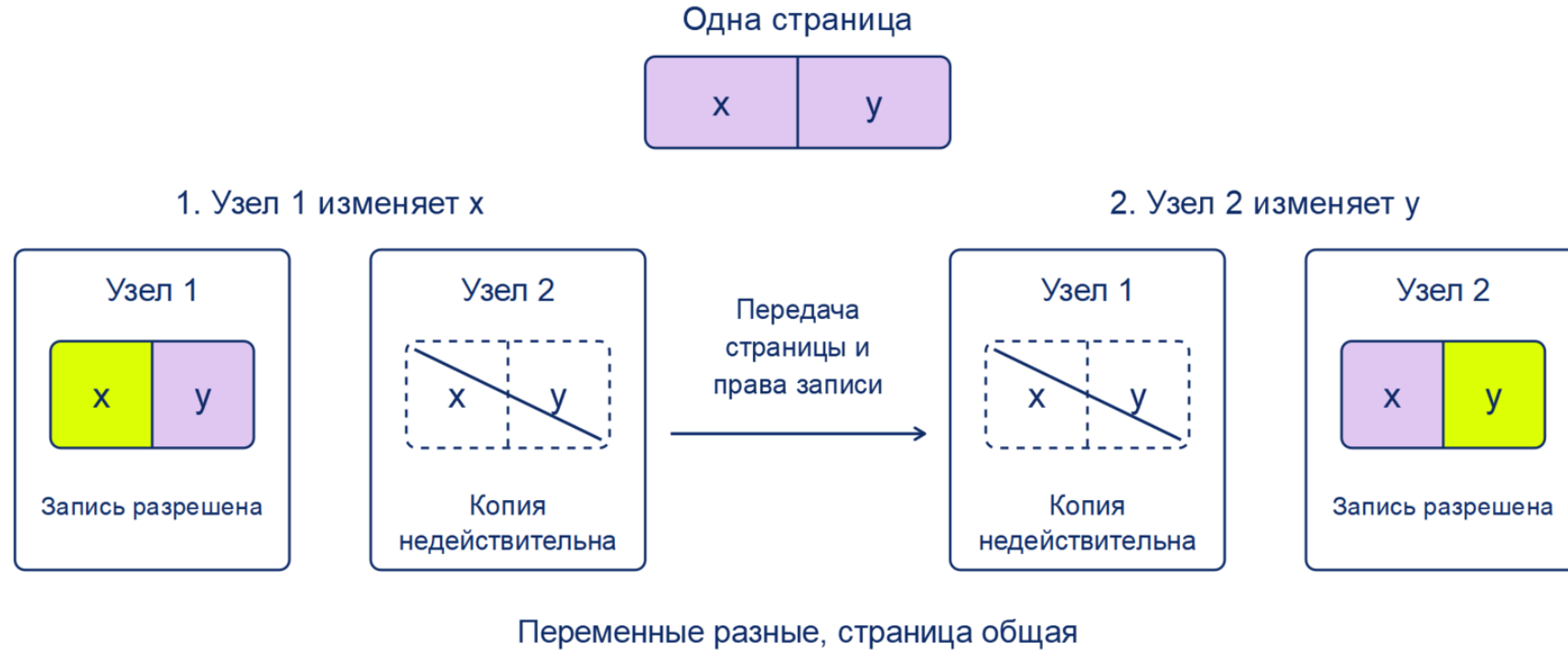
Обновление



- **Инвалидация:** запретить использование старых копий, загрузить актуальные данные при следующем обращении.
- **Обновление:** передать изменения узлам с копиями.

Выбор зависит от того, как часто данные изменяют и читают.

Ложное разделение (false sharing)



- Разные узлы изменяют разные переменные одной страницы.
- DSM управляет доступом к странице целиком.
- Независимые записи вызывают пересылки и инвалидации.

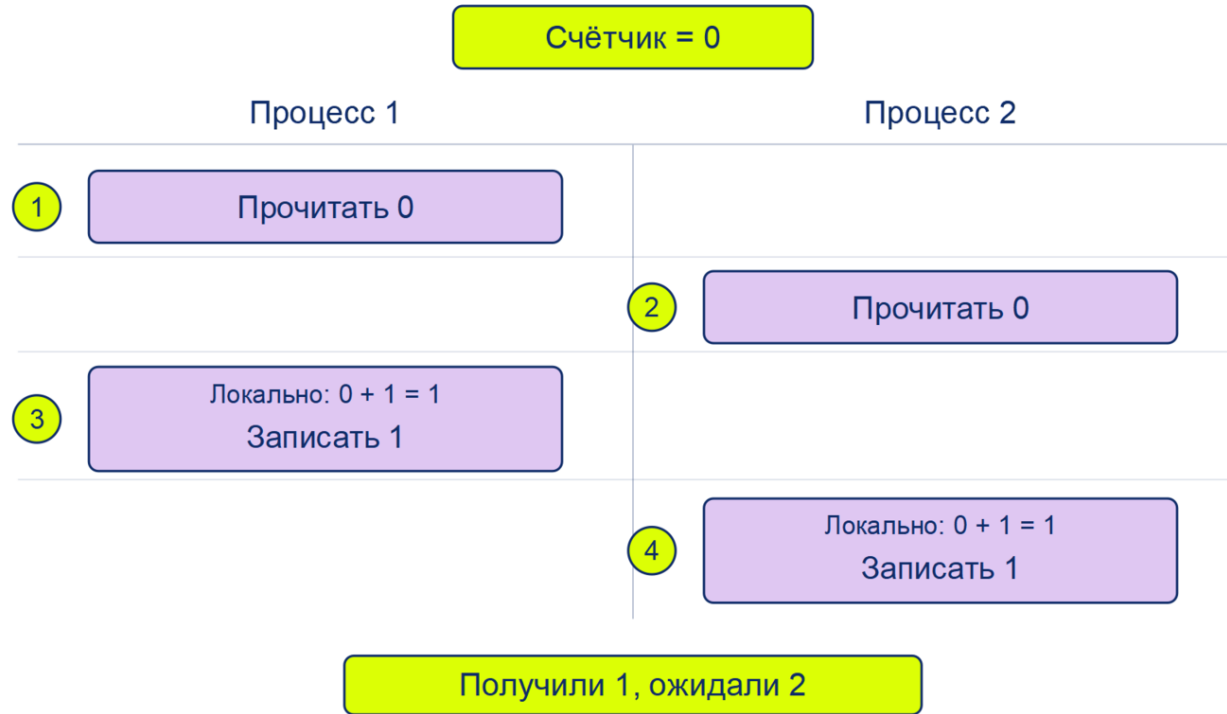
Размещение данных влияет на стоимость доступа.

Размер блока памяти: компромиссы

	Крупные блоки	Мелкие блоки
Передача данных	Меньше запросов при обращении к соседним данным	Меньше ненужных данных
Ложное разделение	Выше вероятность	Ниже вероятность
Учёт копий и прав	Меньше метаданных	Больше метаданных

Размер блока должен соответствовать характеру обращений к памяти.

Общая память не отменяет синхронизацию



- Чтение и запись по отдельности не делают атомарной всю операцию.
- Два процесса могут перезаписать результат друг друга.
- Нужны блокировки или атомарные операции.

Согласованность копий не заменяет синхронизацию приложения.

Цена общей памяти

- Локальное и удалённое обращения имеют разную стоимость.
- Размещение данных и характер обращений определяют объём обмена.
- Совместная работа требует синхронизации.
- Отказы могут сделать общие данные недоступными.

Пространство кортежей



Кортеж – запись из нескольких полей, например ("result", 42, 125).

- Процессы добавляют записи в общее пространство.
- Другие процессы находят их **по шаблону содержимого**.
- Подходящую запись можно прочитать или извлечь из пространства.

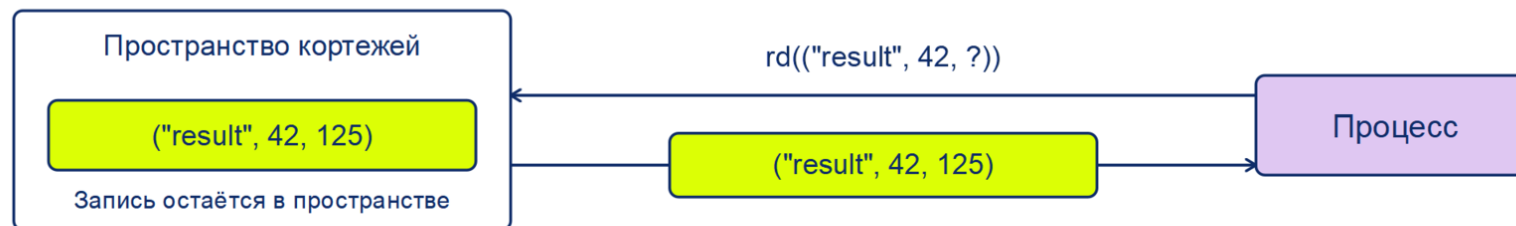
Пример шаблона: ("result", 42, ?) – результат задачи № 42 с любым значением.

Операции с кортежами

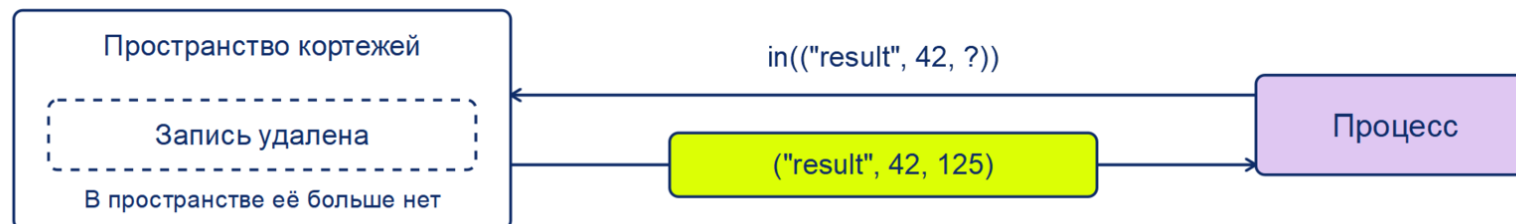
Операция	Действие
out(кортеж)	Добавить запись
rd(шаблон)	Прочитать подходящую запись, оставив её в пространстве
in(шаблон)	Извлечь подходящую запись с удалением

Если подходящей записи нет, rd и in ждут её появления.

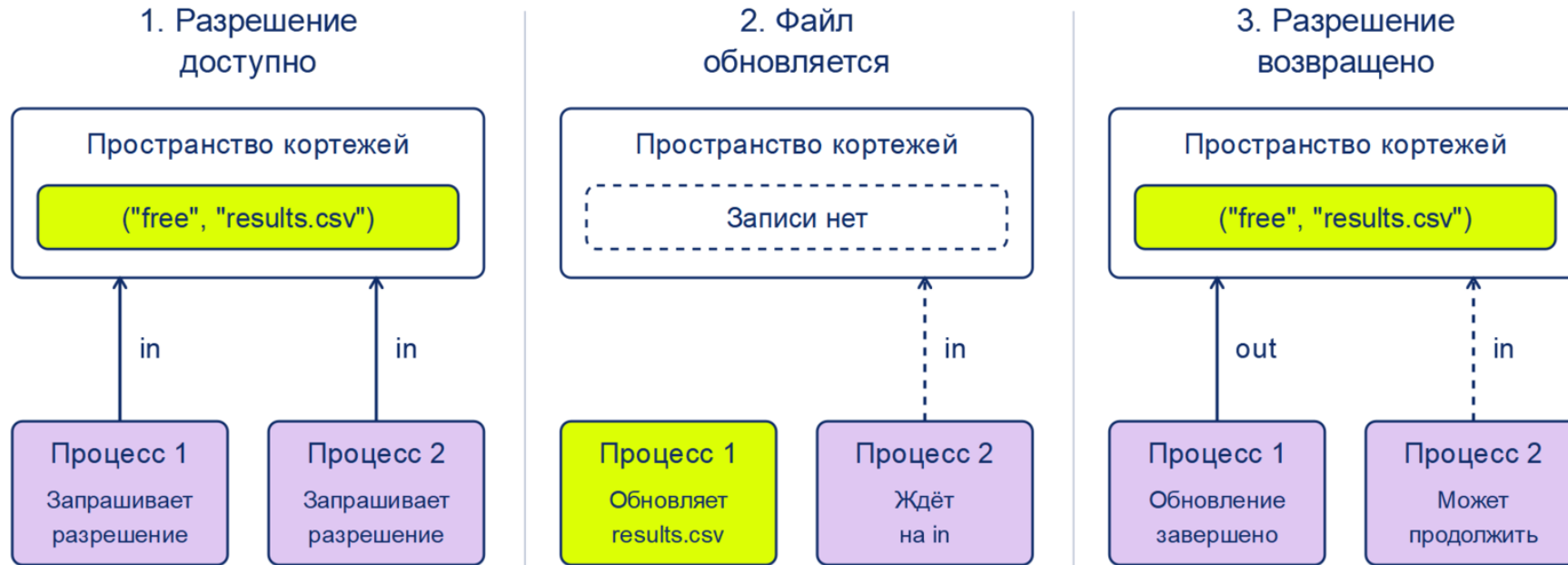
rd — чтение



in — извлечение



Координация через пространство кортежей



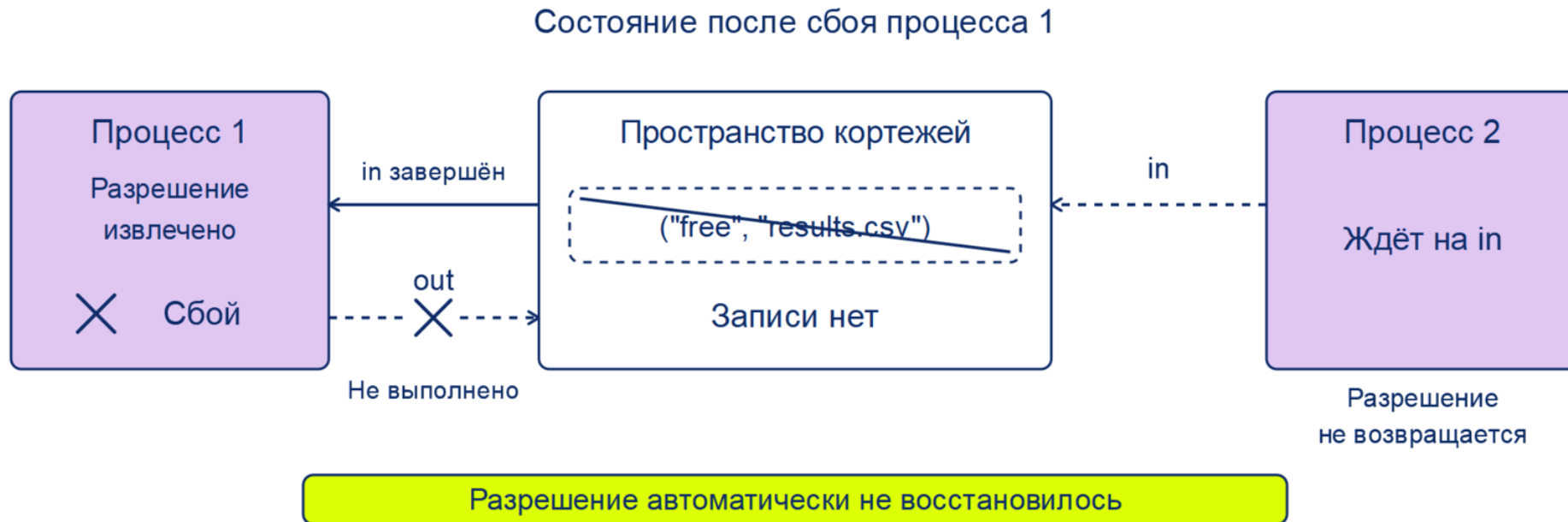
Для обновления файла нужно извлечь единственный кортеж

Одна запись ("free", "results.csv") разрешает обновление общего файла.

- **in** – извлечь запись и получить право обновить файл.
- Пока записи нет, другие процессы ждут.
- **out** – вернуть запись после завершения обновления.

При конкурирующих запросах одну запись получит только один процесс.

Сбой между in и out



- Процесс извлёк разрешение и упал до его возвращения.
- Остальные процессы могут ждать бесконечно.
- Нужно предусмотреть безопасное восстановление разрешения.

Атомарность одной операции не делает атомарной всю последовательность.

Разделяемое состояние на практике

Задача	Пример общего состояния	Механизм
Хранить данные приложения	Заказы и их статусы	База данных
Повторно использовать данные	Копии карточек товаров	Распределённый кеш
Координировать процессы	Участники и их роли	Сервис координации

Важно, какие операции атомарны и когда изменения видны другим.

Какую модель выбрать?

- **Конвертация видео.** Распределить обработку загруженных роликов между свободными исполнителями.
- **Просмотры видео.** События нужны сервису рекомендаций и статистике авторов. Нужна возможность пересчитать статистику за прошлый месяц.
- **Общая конфигурация.** Все экземпляры сервиса используют общие настройки; новый экземпляр должен получить текущее состояние.

Кто получает данные? Нужно ли хранить историю или текущее состояние?

Выбор модели взаимодействия

Что требуется	Подходящая абстракция
Распределить работу между исполнителями	Очередь задач
Передать событие независимым подписчикам	Pub/sub
Сохранить историю для повторного чтения	Журнал
Читать и изменять общие записи	Общее хранилище
Обращаться к общим данным как к памяти	DSM
Находить и извлекать записи по шаблону	Пространство кортежей

Модели можно сочетать. Гарантии и свойства реализации нужно уточнять отдельно.

Что проверить после выбора модели

- **Отказы:** что сохранится и как продолжится работа?
- **Подтверждения:** что именно считается выполненным?
- **Повторы:** безопасно ли выполнить операцию ещё раз?
- **Совместная работа:** какие действия атомарны, какой порядок гарантирован?
- **Перегрузка:** что произойдёт при исчерпании ресурсов?

Материалы

- [Distributed Systems: Concepts and Design](#) (разделы 6.1, 6.3-6.5)
 - плюс глава [Distributed Shared Memory](#)
- [Distributed Systems](#) (разделы 2.1, 4.3)
- [The Many Faces of Publish/Subscribe](#)
- [RabbitMQ vs Kafka Series](#)