

# Гарантии доставки сообщений в YDB Topics

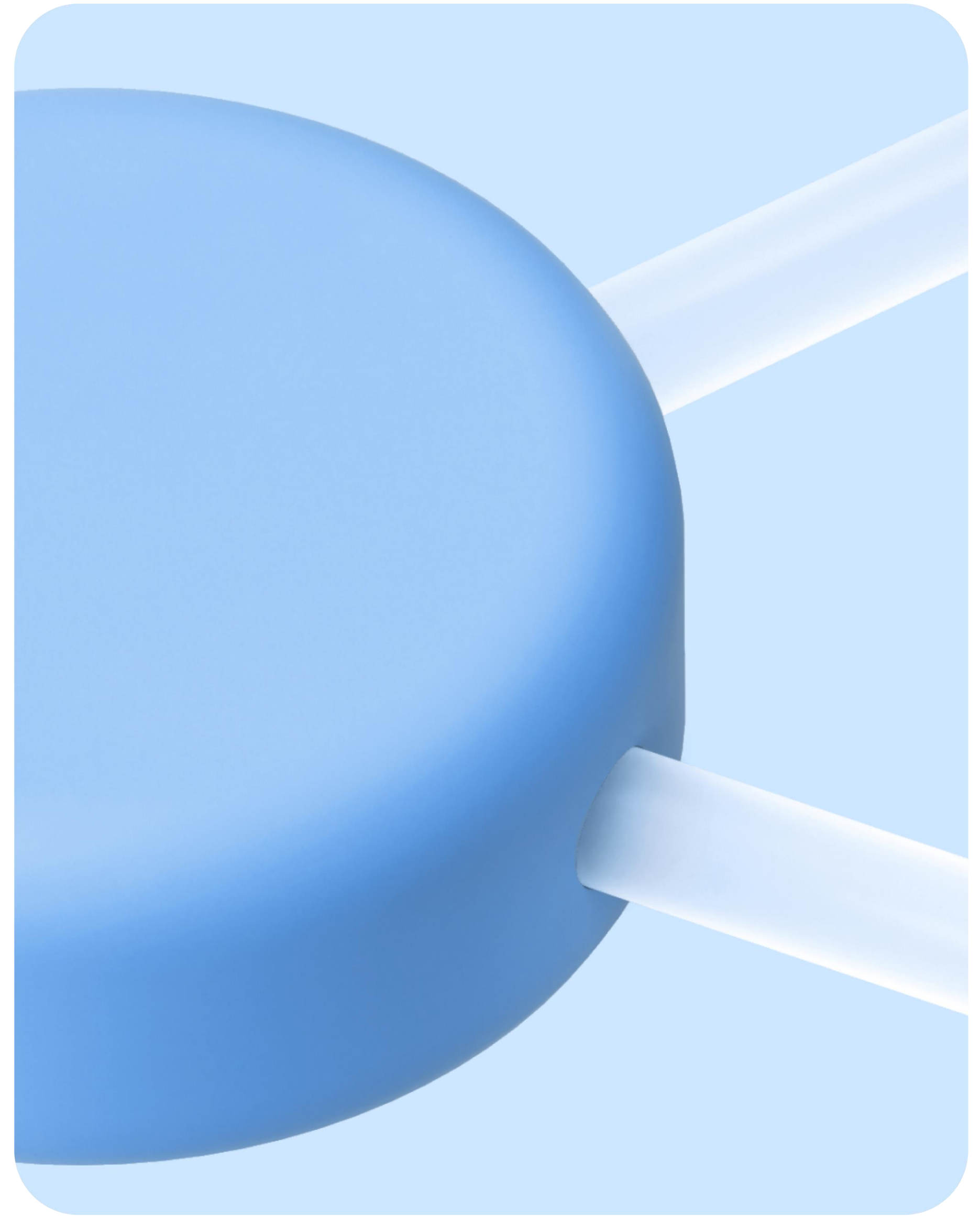
**Зевайкин Александр**

Руководитель группы разработки,  
кандидат технических наук, доцент

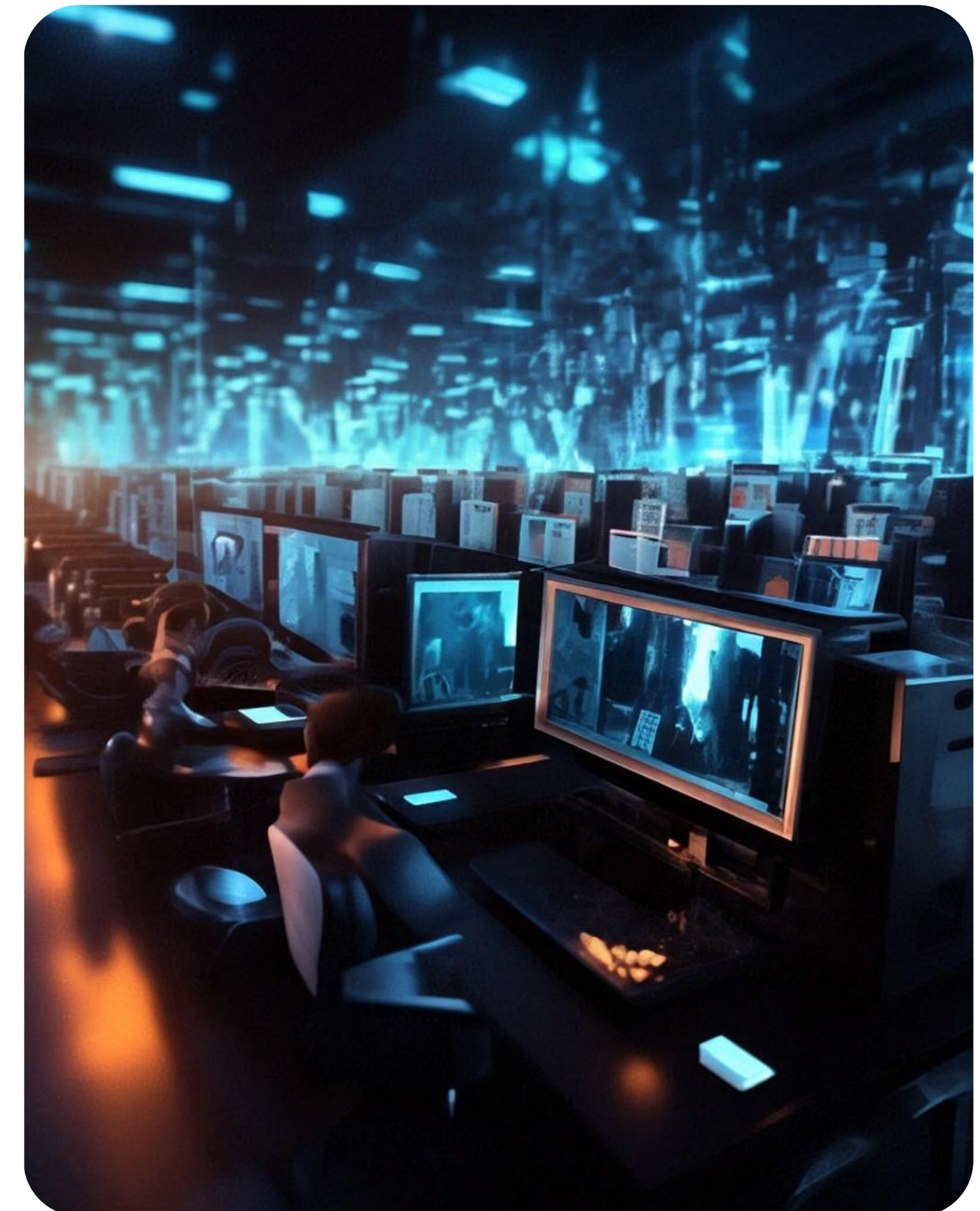


**Saint  
HighLoad++**

**Проблема**



# Сети ненадёжны



# Потери на прикладном уровне сети

## Надёжная сеть

- Сообщение принимается тогда и только тогда, когда оно отправлено. Порядок может меняться.

# Потери на прикладном уровне сети

## Надёжная сеть

- Сообщение принимается тогда и только тогда, когда оно отправлено. Порядок может меняться.

## Сети с обычными потерями

- Сообщения могут быть потеряны, продублированы
- Если повторять попытки, сообщение в конечном итоге будет передано


# Потери на прикладном уровне сети

## Надёжная сеть

- Сообщение принимается тогда и только тогда, когда оно отправлено. Порядок может меняться.

## Сети с обычными потерями

- Сообщения могут быть потеряны, продублированы
- Если повторять попытки, сообщение в конечном итоге будет передано



Повторы,  
дедупликация

# Повторы

Пример, когда обязаны завершить операции:

Доставлено  
сообщение  
«пользователь  
заказал товар»



Доставлено  
сообщение  
«пользователь  
оплатил»



Нестабильная  
сеть — потеряно  
сообщение  
«передать товар  
в доставку»



Последнее сообщение нужно повторять множество раз

# Борьба с дублями

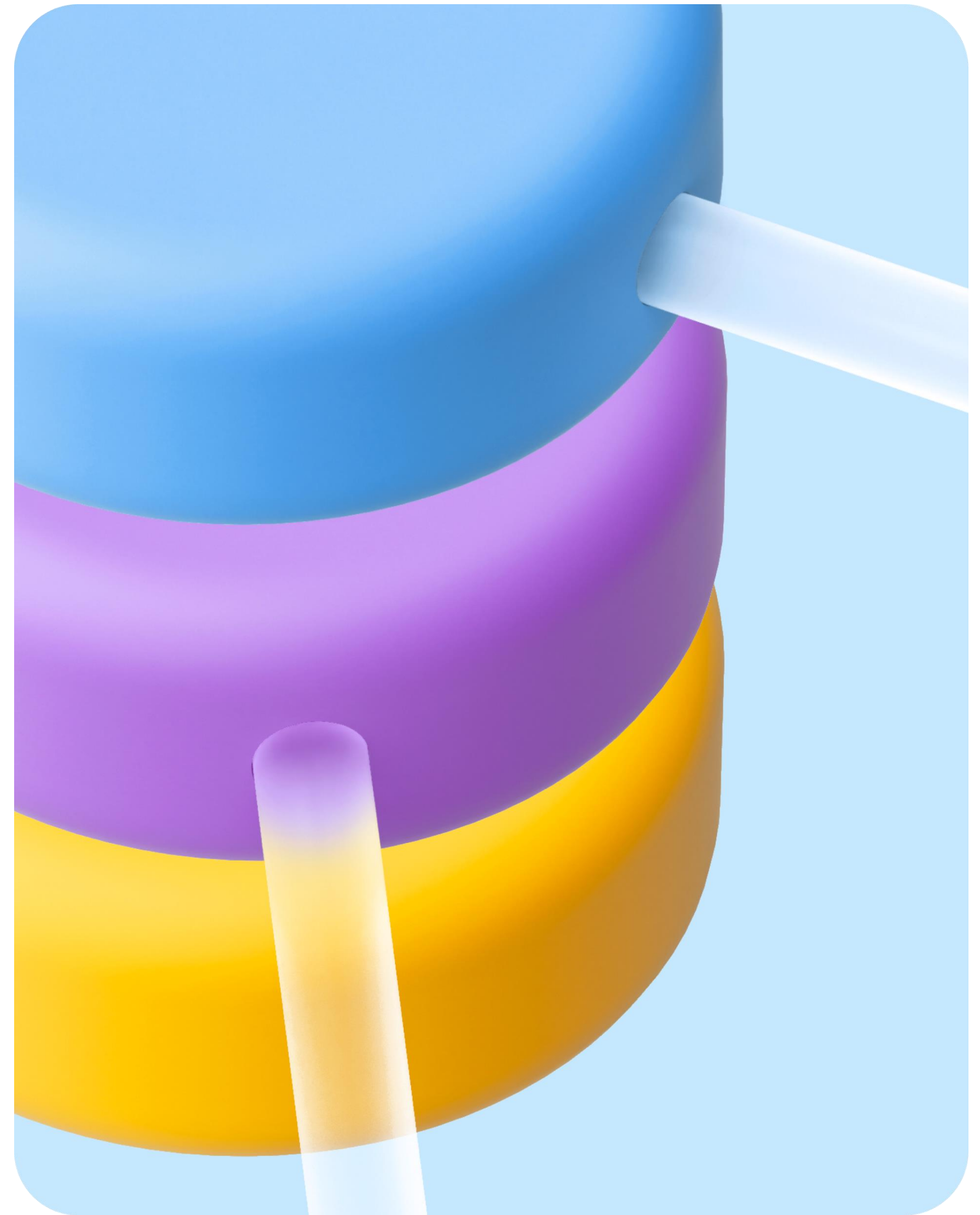
Начали повторять  
сообщение  
«передать товар  
в доставку»

Могут быть дубли

Нужно  
дедуплицировать



# **Базовые понятия гарантий доставки**



# At-most-once 1

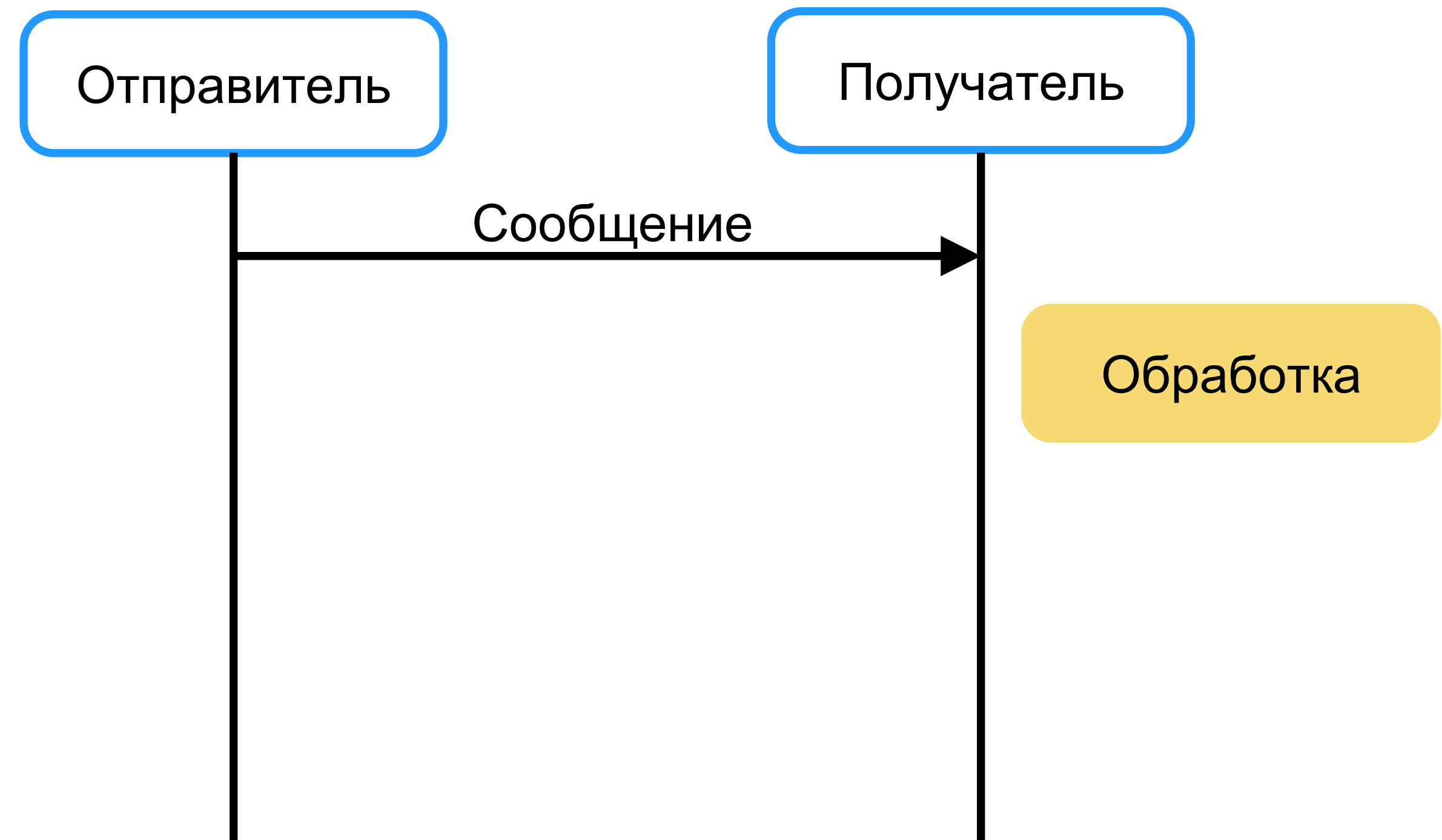
«Расслабленная доставка»

## Отправитель

- Не ждёт подтверждений
- Игнорирует любые ошибки

## Получатель

- Подтверждение не отправляет
- Игнорирует любые ошибки



Пример: периодическая отправка состояния, отправка координат автомобиля в движении

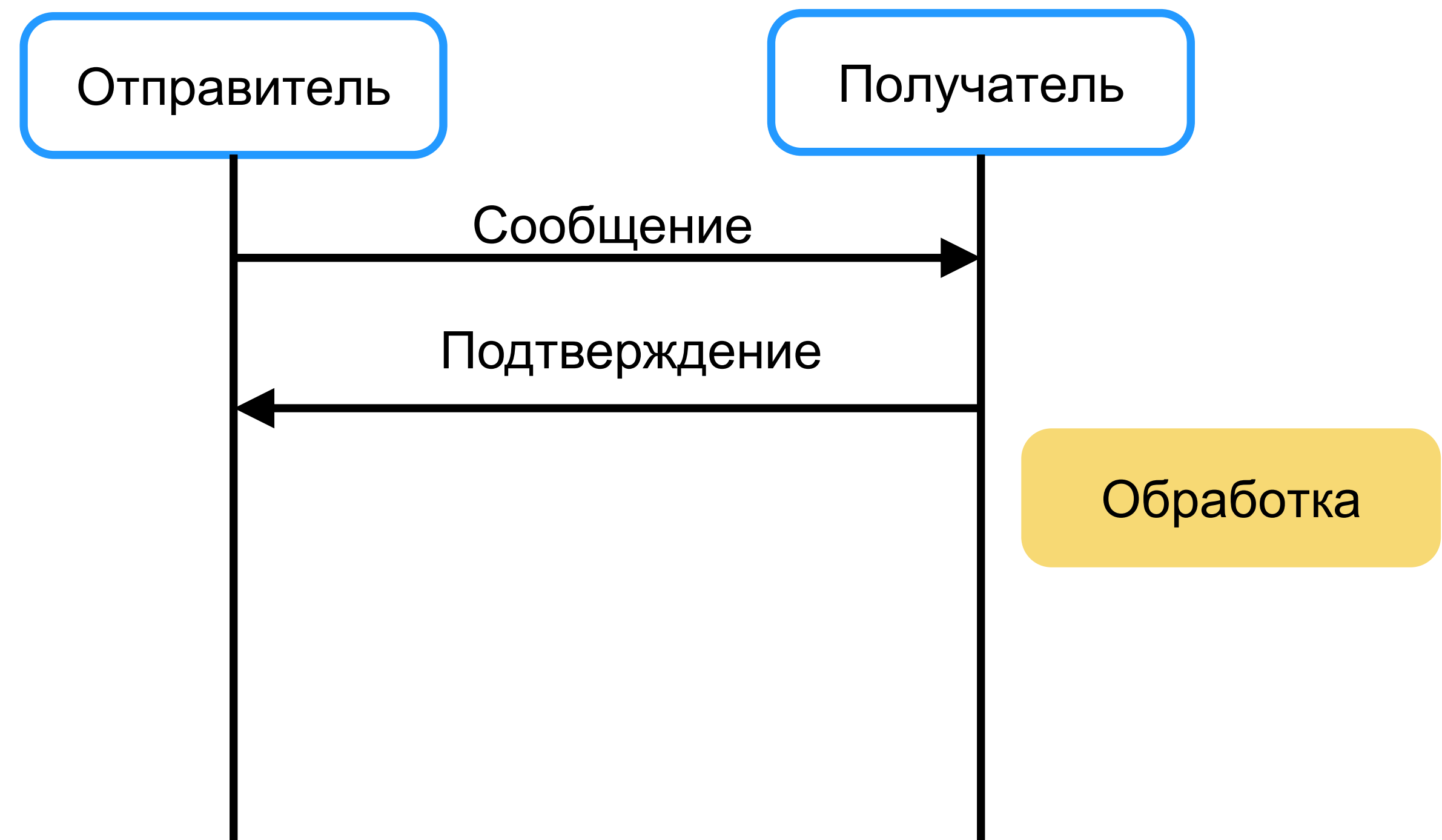
# At-most-once 2

## Отправитель

- Принимает подтверждения

## Получатель

- Подтверждение отправляет до обработки



Пример: отправка координат автомобиля учитывает подтверждение для планирования следующей отправки

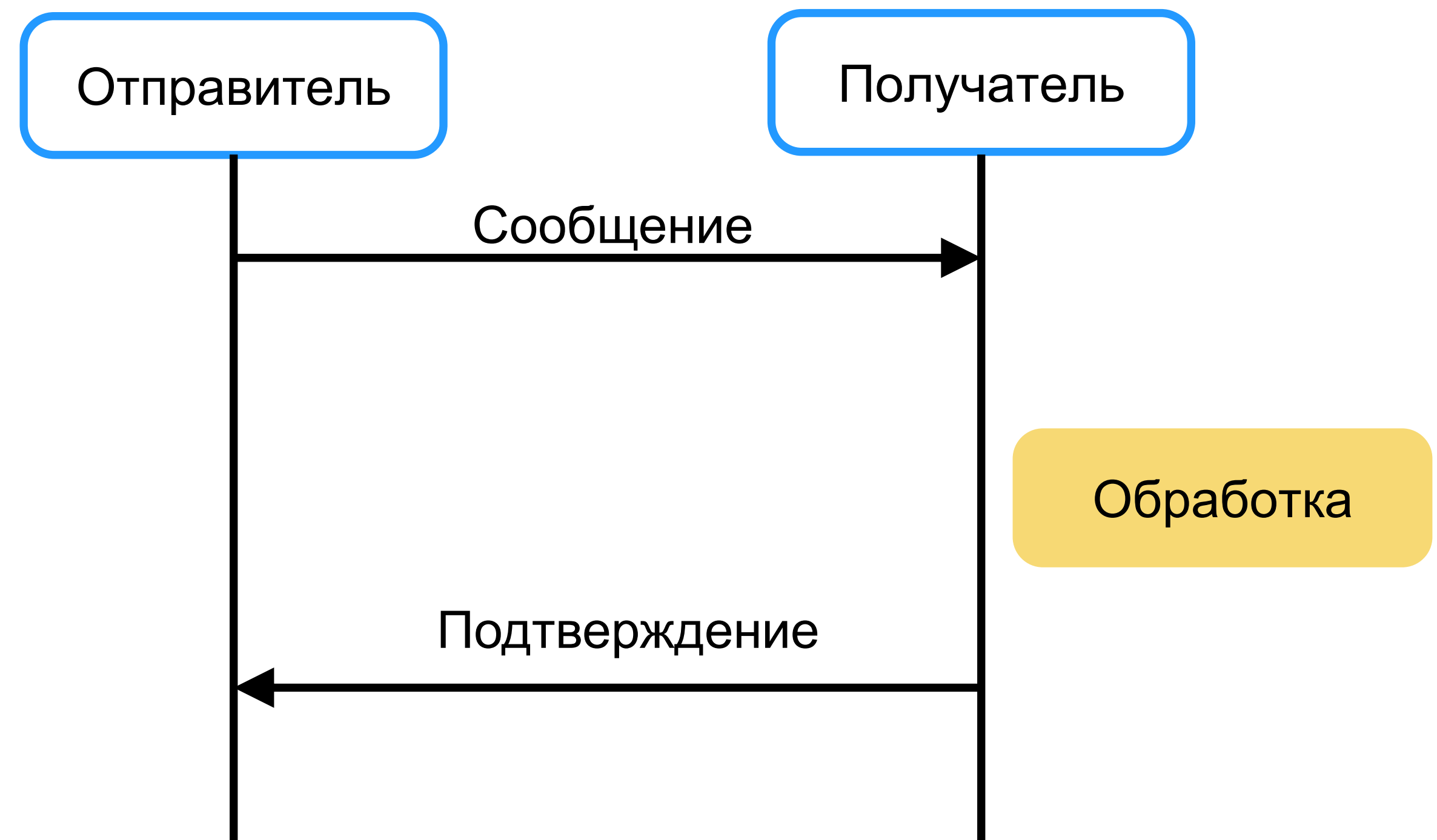
# At-least-once

## Отправитель

- Ждёт подтверждений
- Есть тайм-аут ожидания
- Если подтверждения нет, то повтор сообщения

## Получатель

- Сначала обработка сообщения, потом отправка подтверждения



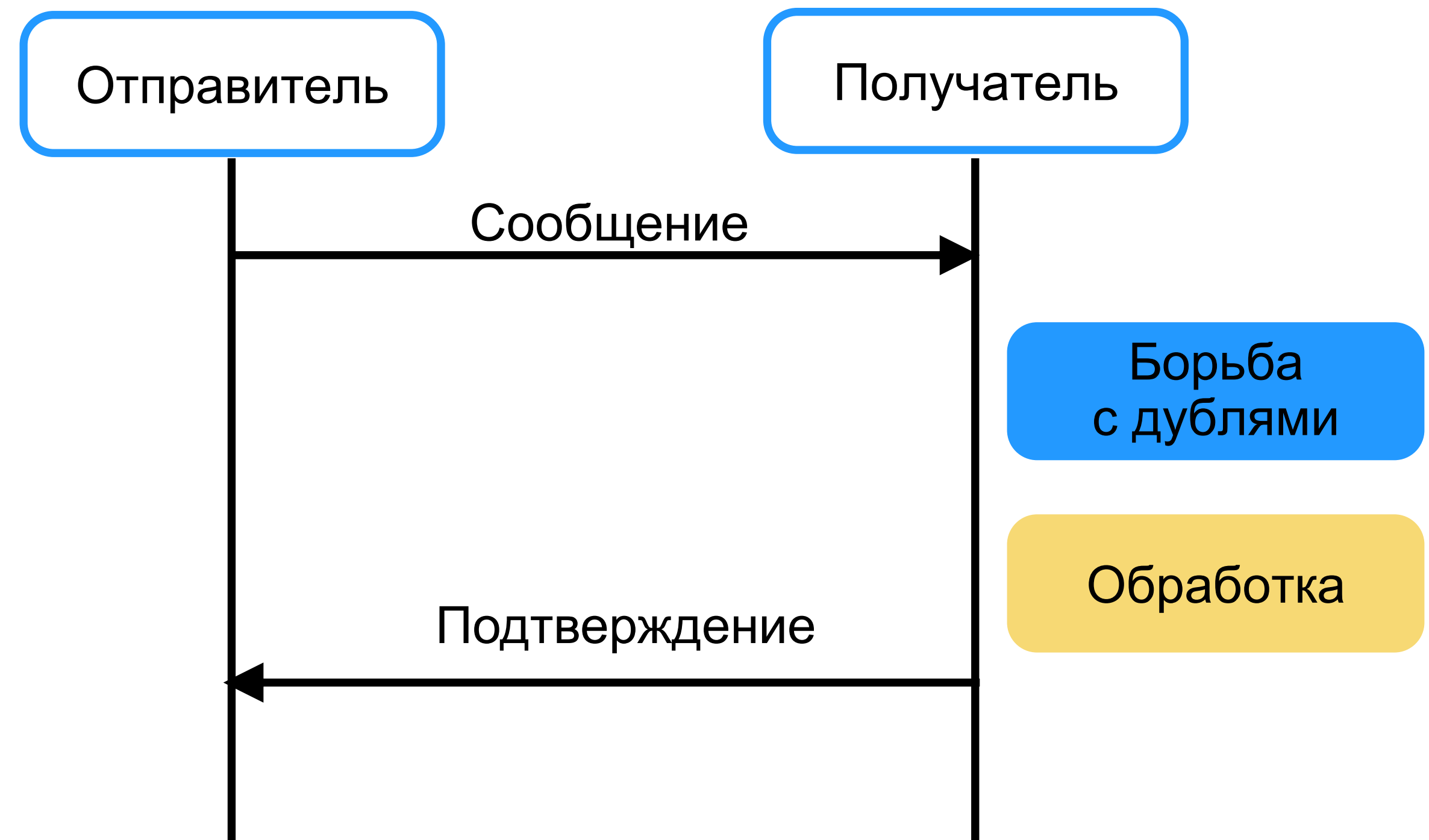
# Exactly-once

## Отправитель

- Всё, как в at-least-once

## Получатель

- Всё, как в at-least-once
- Плюс борьба с дублями



Примеры: денежные переводы

# Борьба с дублями

**Идемпотентная  
операция**

**Удаление дублей**

# Идемпотентная операция

Идемпотентность – свойство объекта или операции при повторном применении операции к объекту давать тот же результат, что и при первом

## Примеры из математики

### Сложение с нулем

$$a = a + 0 = a + 0 + 0 = \dots$$

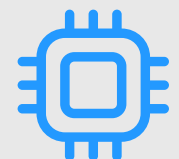
### Нахождение максимума

$$\begin{aligned} \max(a,b) &= \max(\max(a,b),b) = \\ \max(a, \max(a,b)) &= \dots \end{aligned}$$



## Примеры из информатики

UPSERT-запрос (SQL)



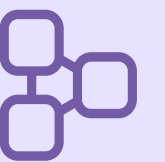
# Удаление дублей

**Вместе с сообщениями  
передавать ключ  
идемпотентности**



**Сохранять в СУБД с проверкой  
уникальности**

Ошибка при вставке дубля

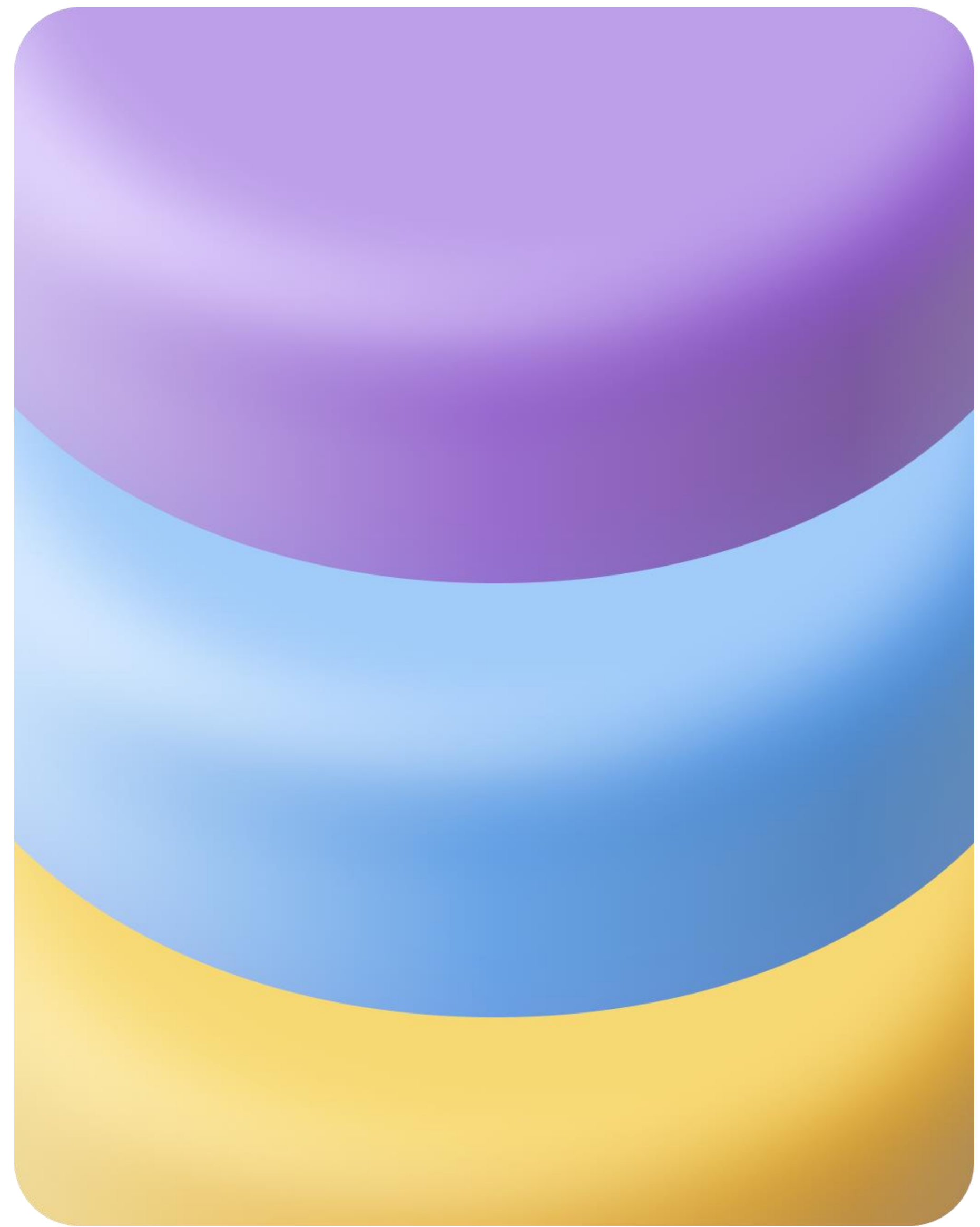




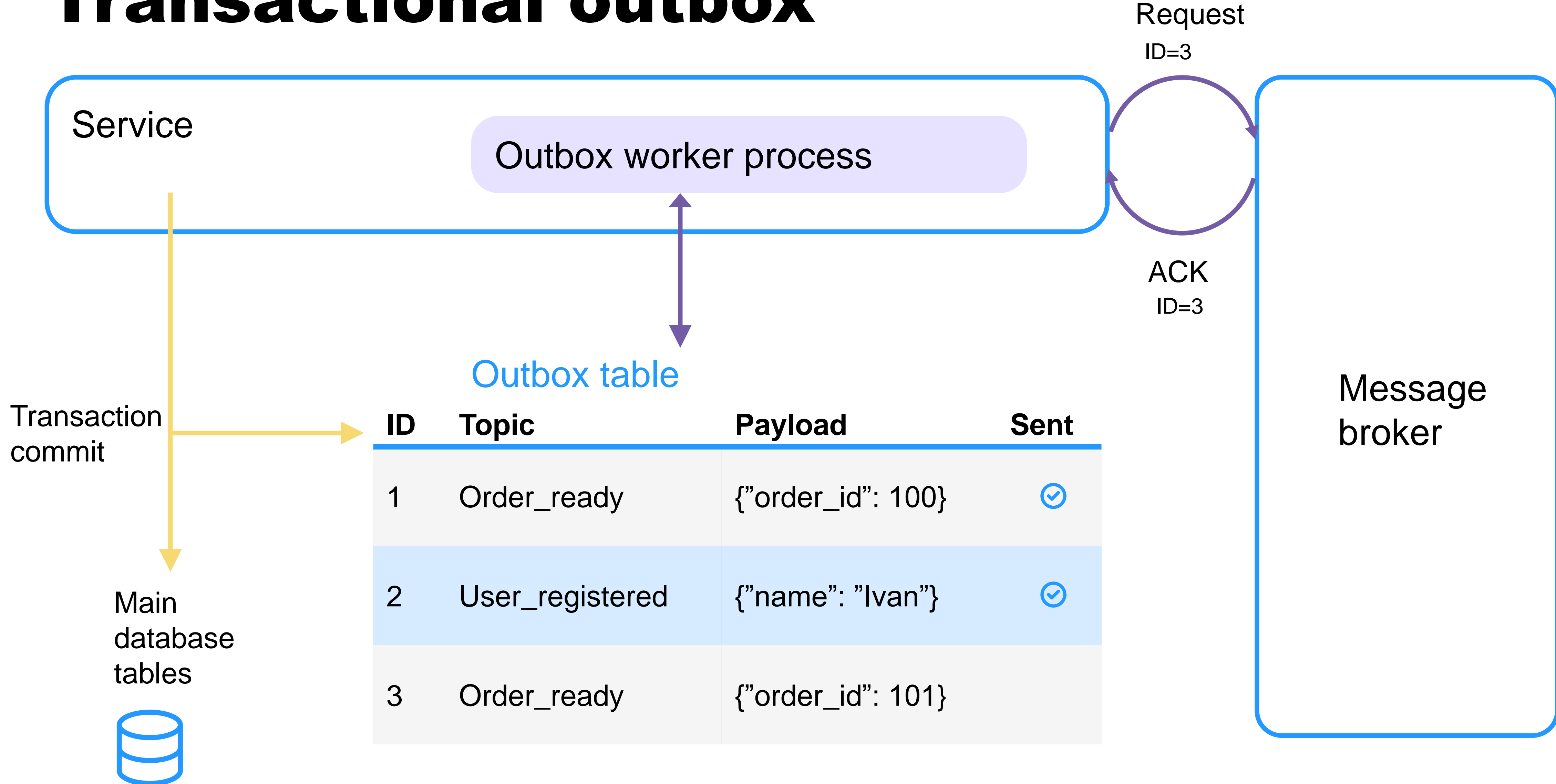
# Сравнение гарантий

Гарантия	Сообщений доставлено	Пропуски ВОЗМОЖНЫ	Дубли ВОЗМОЖНЫ
At-most-once	0,1	+	-
At-least-once	1+	-	+
Exactly-once	1	-	-

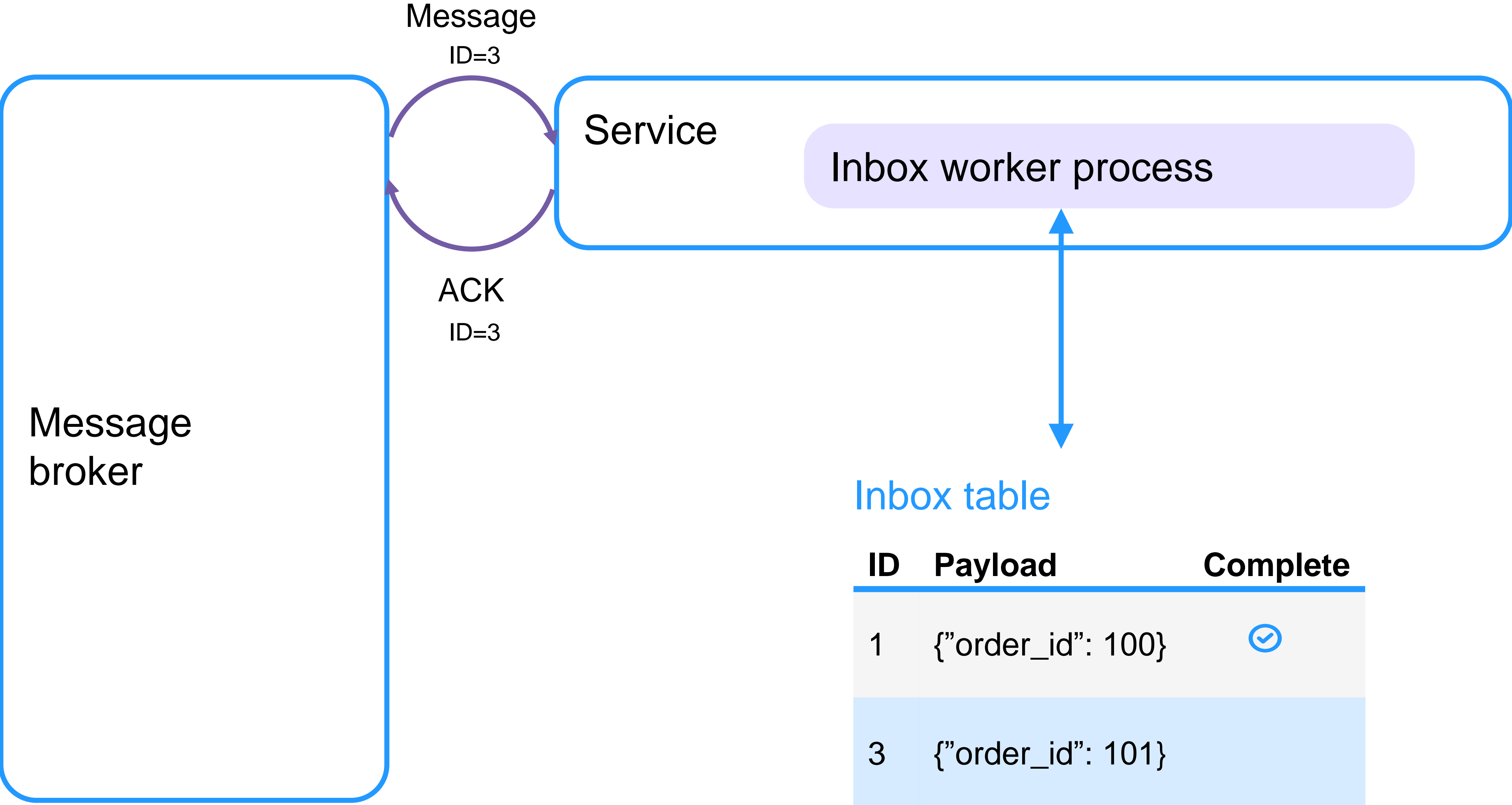
# Паттерны микросервисной архитектуры



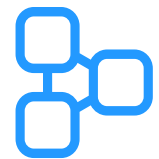
# Transactional outbox



# Transactional inbox



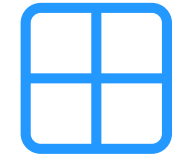
# Недостатки Transactional Outbox



**Требует транзакционной базы данных**



Увеличенная латентность



**Таблица outbox — лишняя нагрузка**



Нужно писать сложный код outbox worker

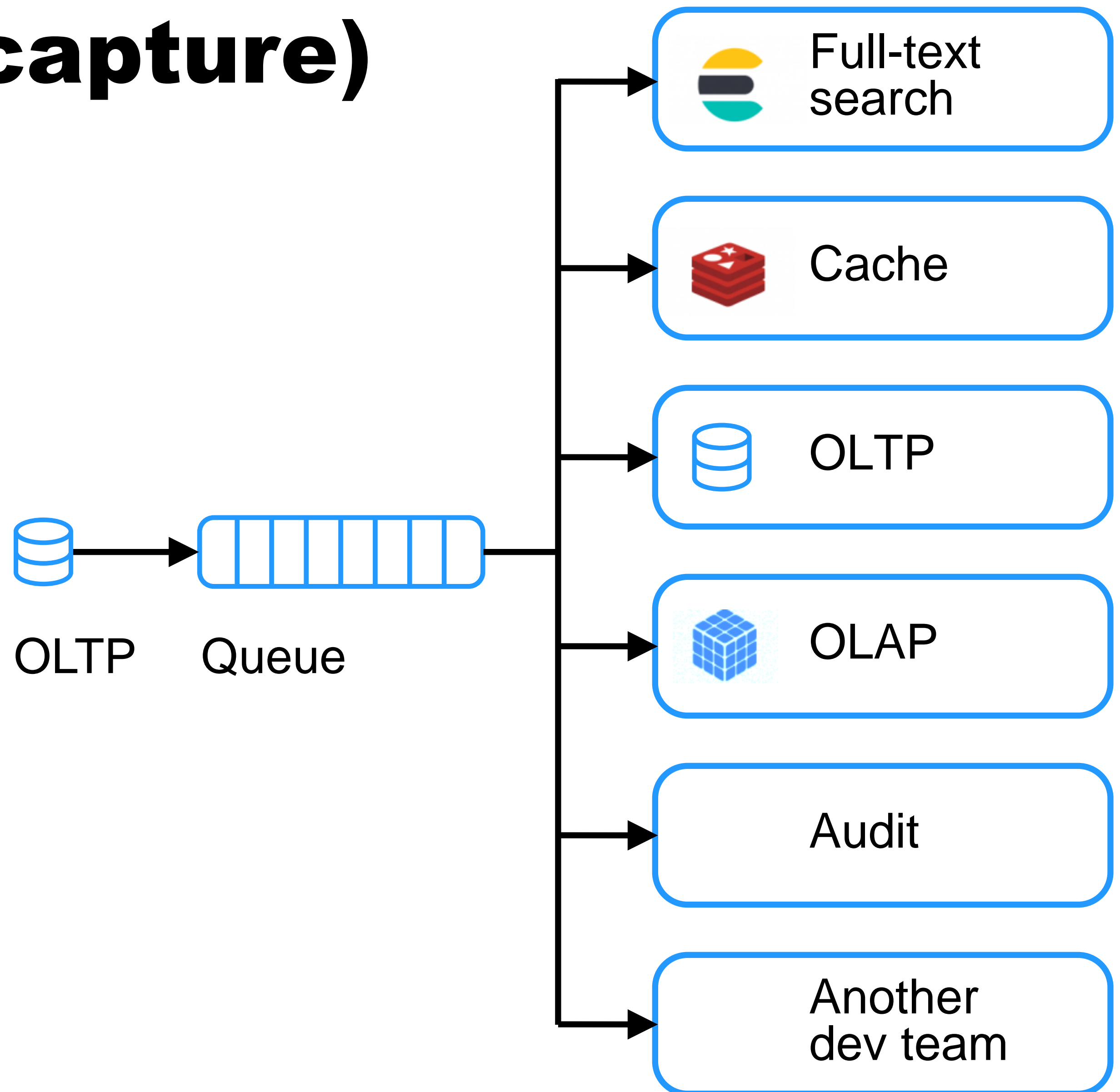


**Число записей нужно контролировать**

Стоит обратить внимание на CDC

# CDC (change data capture)

- Захват изменений источника
- Формирует поток изменений
- Сохранение изменений в получателе
- Не требуется транзакционная база данных
- Готовые фреймворки, например, Debezium
- Поддержка в современных СУБД



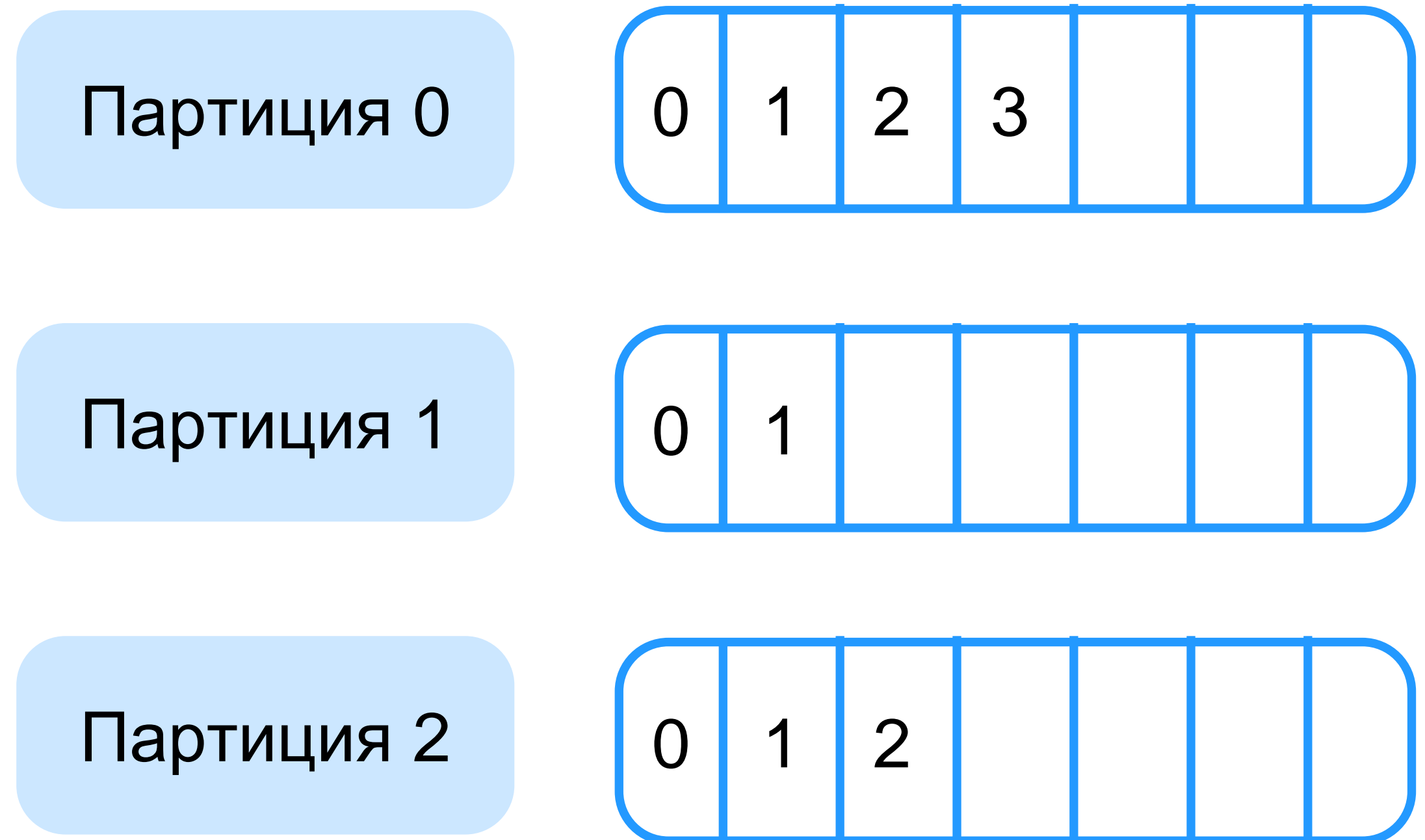
**Kafka**



# Сущности Kafka

- Пользовательские данные сгруппированы по **топикам** (topics)
- Топик разделён на **партиции** (partitions)
- Одна партиция — это распределённый лог **сообщений**
- Номер сообщения в партиции — **смещение** (offset)

## Топик А





# Подтверждения в Kafka: `acks = 0`

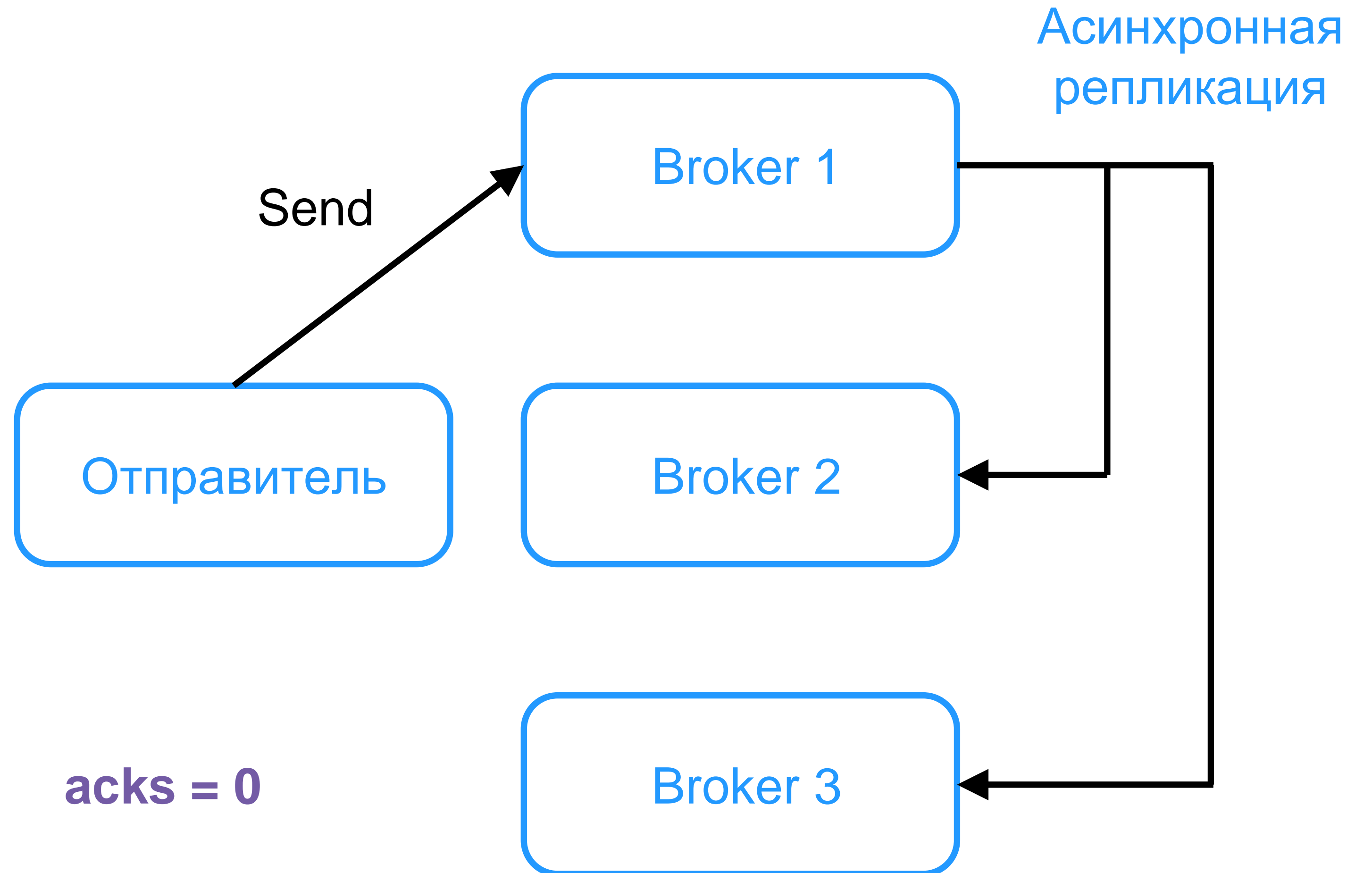
## Отправитель

- Шлёт сообщение
- Не ожидает подтверждения

## Брокер

- Не отвечает
- Асинхронно реплицирует

At-most-once



# Подтверждения в Kafka: acks = 1

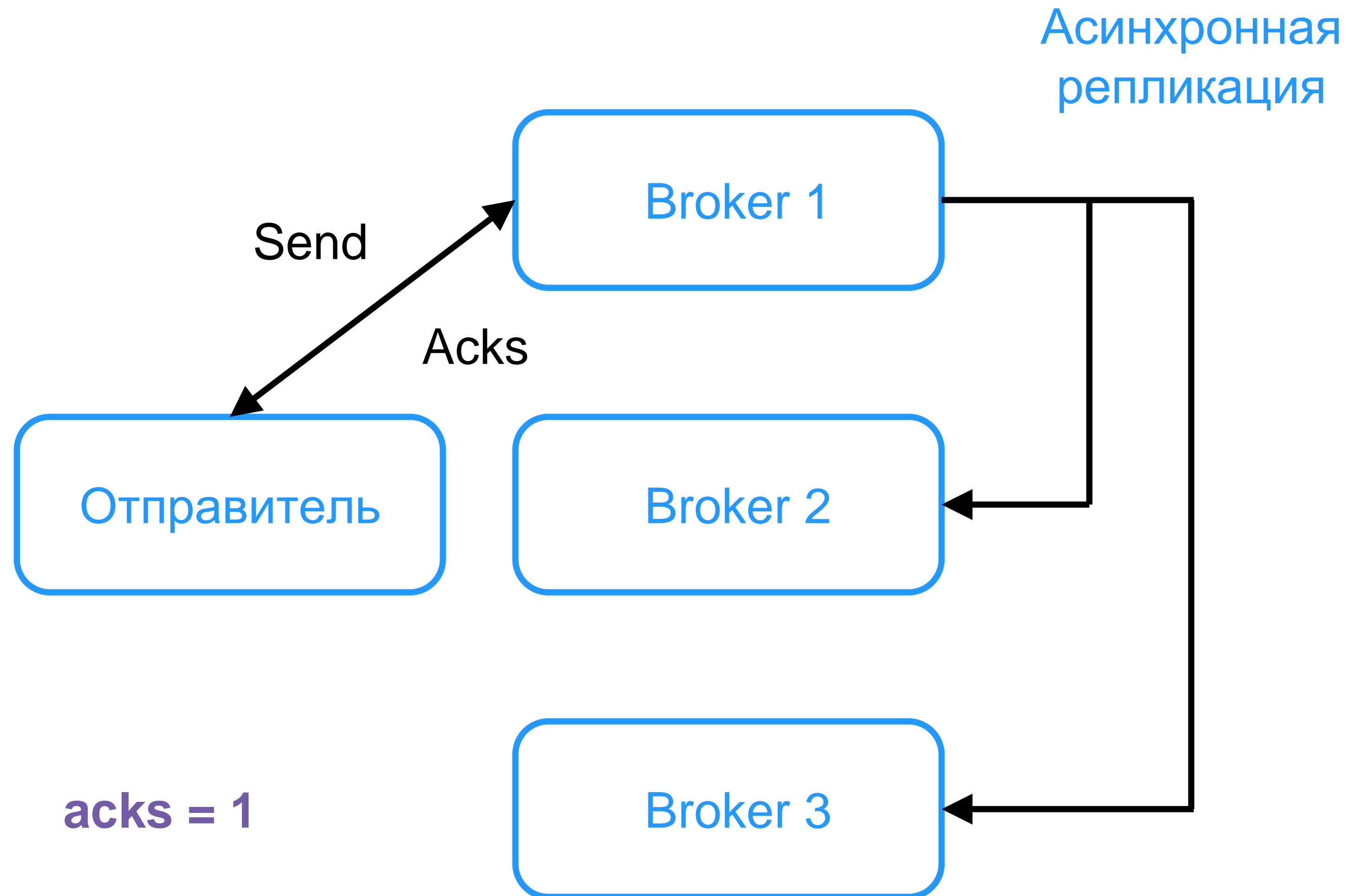
## Отправитель

- Шлёт сообщение
- Ожидает 1 подтверждения

## Брокер

- Не отвечает
- Асинхронно реплицирует

At-least-once с лидером



# Подтверждения в Kafka: acks = all

## Отправитель

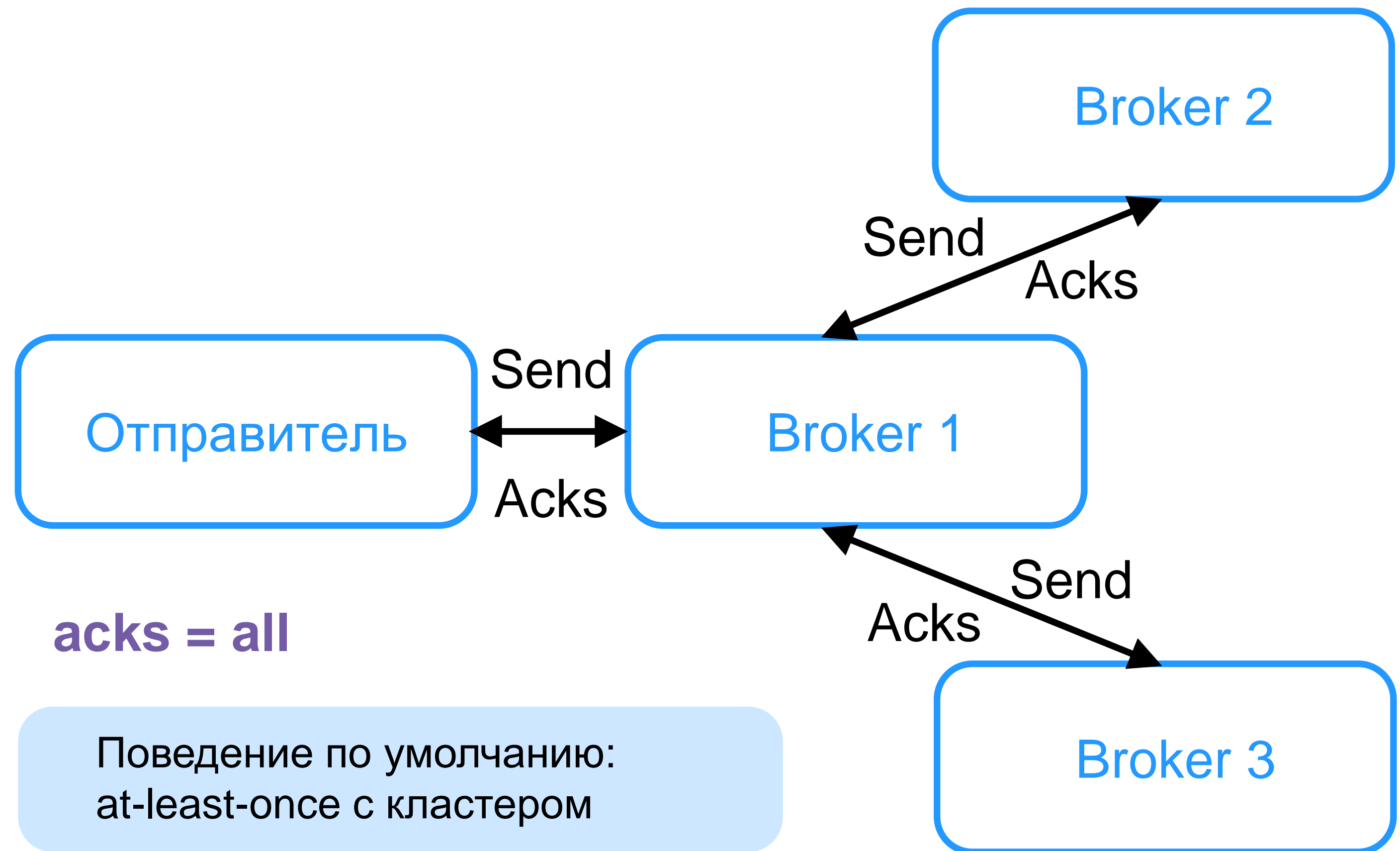
- Шлёт сообщение
- Ожидает всех подтверждений

## Брокер

- Синхронно реплицирует
- Отвечает

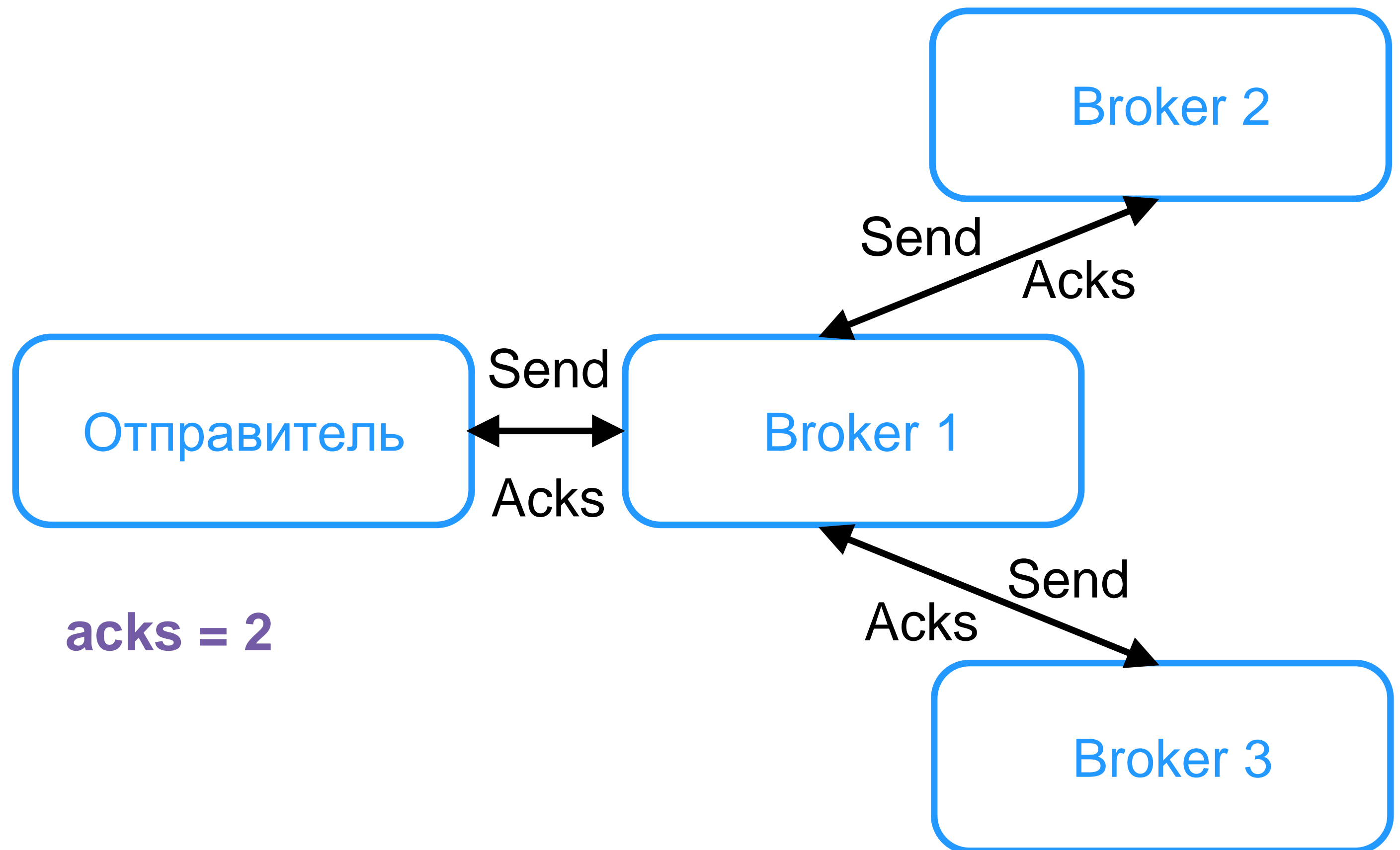
Максимальные гарантии  
отсутствия потерь

Низкие гарантии доступности  
на запись



# Подтверждения в Kafka: acks = 2

Переживаем потерю одного брокера



# Идемпотентный отправитель

## Поля в сообщениях

- Producer id
- Sequence number

## Флаг отправителя

Idempotence = true

## Инвариант брокера

Sequence number строго на 1 больше для пары:  
(producer id, partition)

При разрыве соединения сбрасывается в ноль sequence number

Kafka: We can only guarantee idempotent production within a single producer session

# Сброс счётчика

Счётчик **переживает** только разрыв сети между отправителем и брокером

Счётчик **не переживает** перезапусков отправителя или брокера, смену лидера, обновление Kafka

KIP-98

We can only guarantee idempotent production within a single producer session

# **Что делать отправителю при перезапуске**

**Ничего**

**Посмотреть записанное**

**Хранить смещения**

# Что делать отправителю при перезапуске

## Ничего

Получатель должен бороться  
с дублями

- Удаление дублей
- Идемпотентная обработка

Посмотреть записанное

Хранить смещения



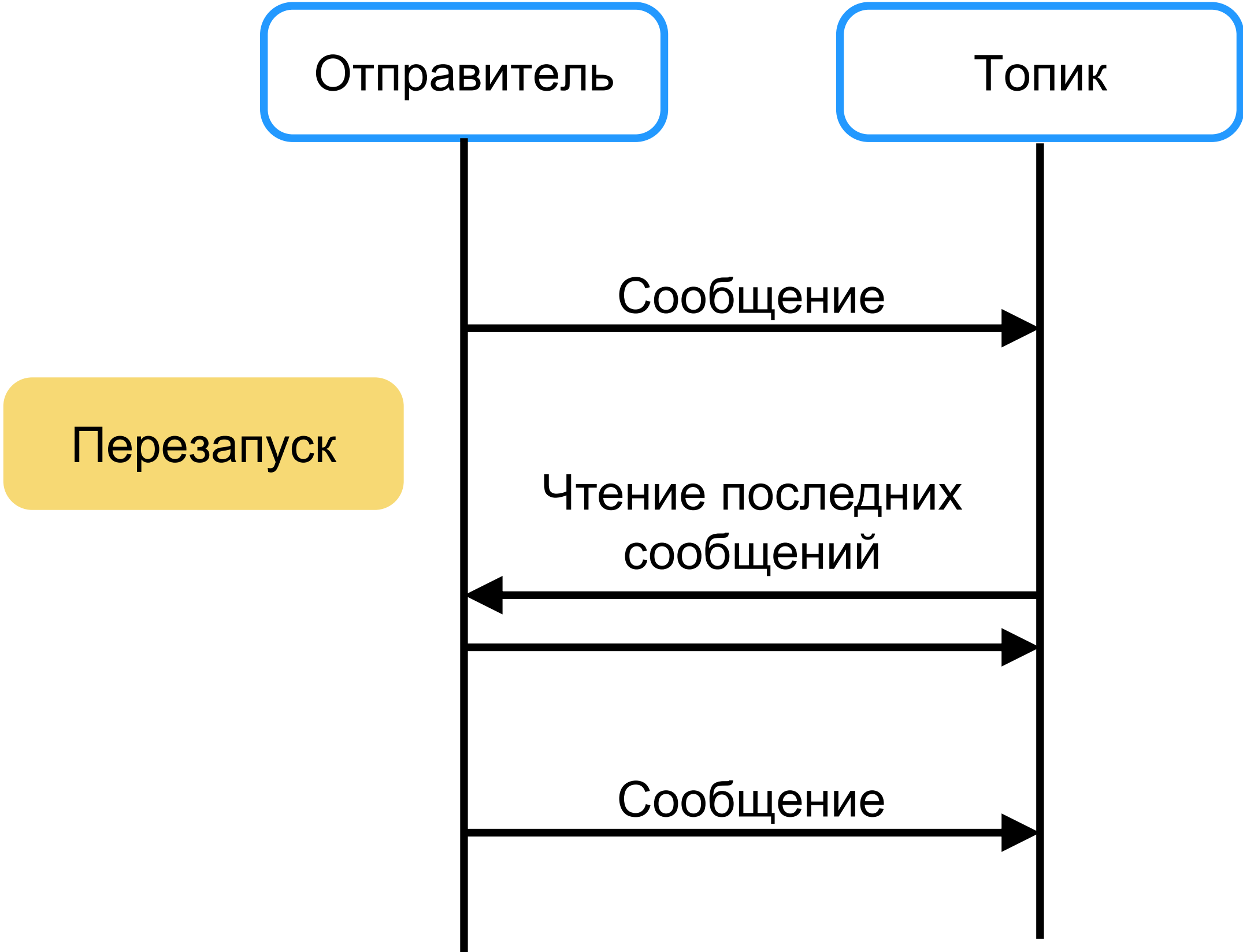
# Что делать отправителю при перезапуске

Ничего

## Посмотреть записанное

- Вычитать последние сообщения
- Проверить ключи идемпотентности

Хранить смещения



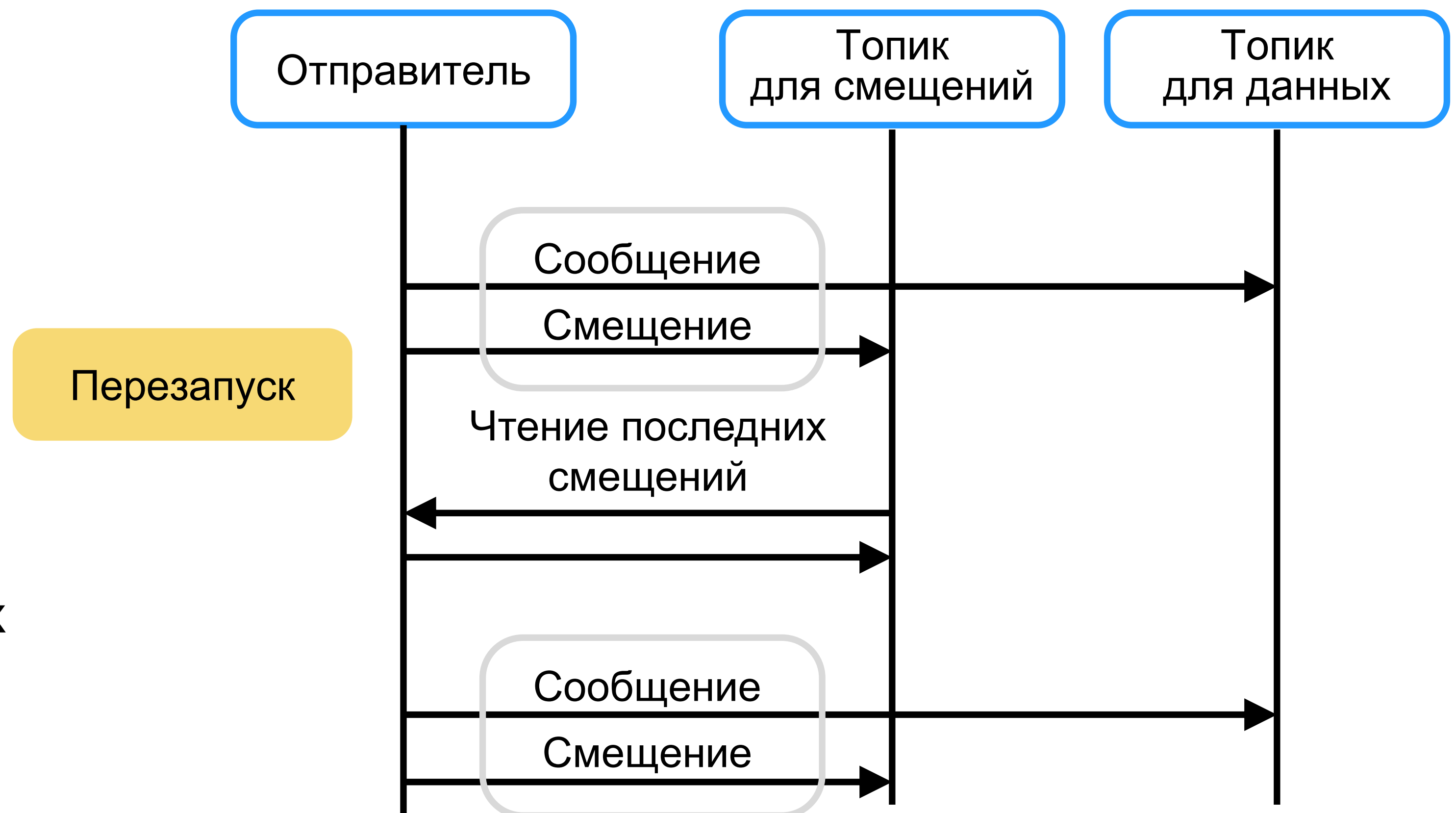
# Что делать отправителю при перезапуске

Ничего

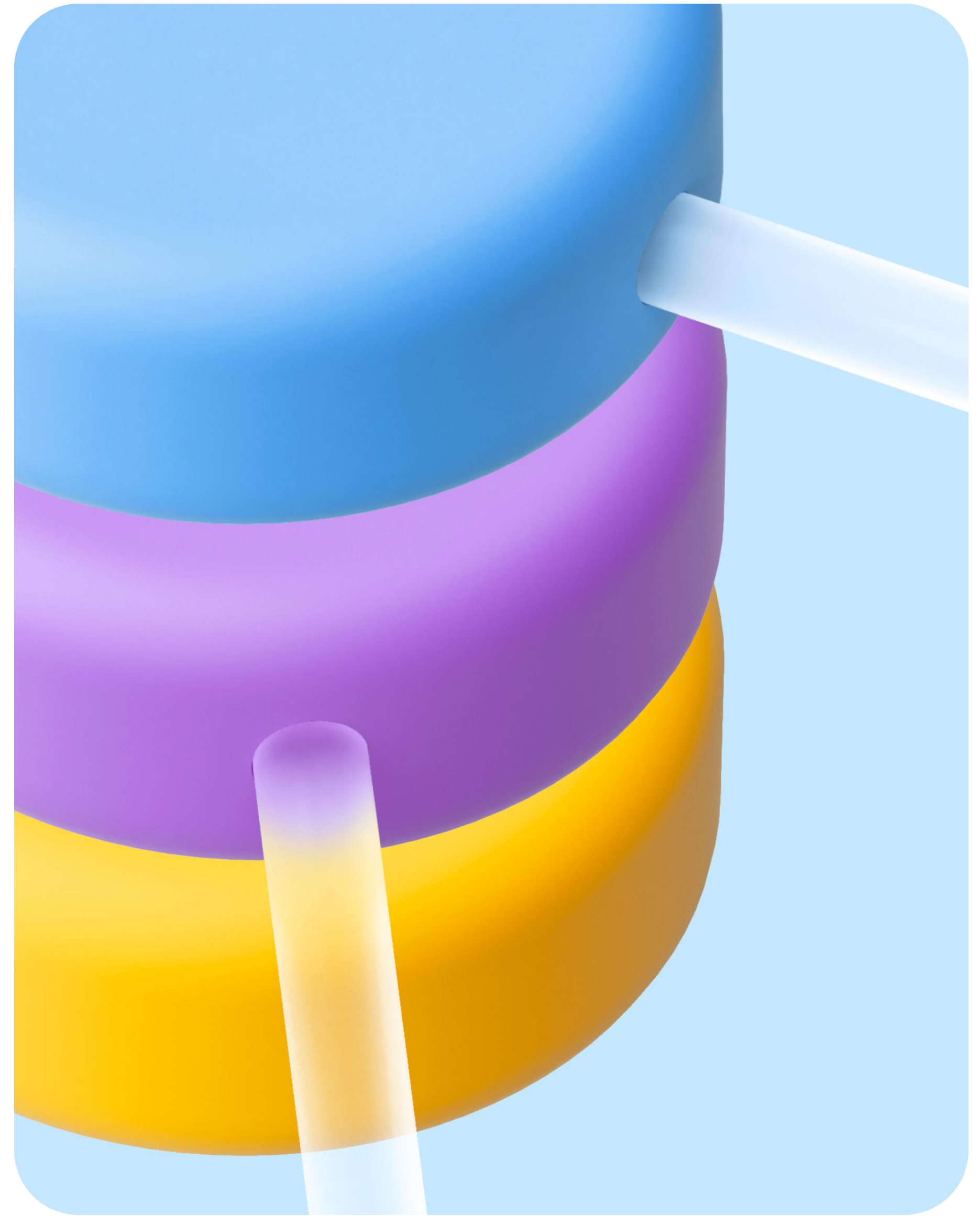
Посмотреть записанное

## Хранить смещения

- Писать в транзакции в 2 топика
- Вторым топиком — сжатый, для смещений в топике данных
- Быстро прочитать последнее смещение



# YDB Topics



# Что такое YDB Topics

Масштабируемый сервис  
для надёжной передачи  
упорядоченных потоков  
сообщений



**2013**

**2017**

**2022**

Production  
Яндекса  
поверх Kafka

Production  
Яндекса  
поверх YDB

Выведен  
в опенсорс

# Платформа YDB

- Распределённый консенсус
- Распределённый слой хранения данных
- Горизонтальное масштабирование, вплоть до тысяч узлов
- Катастрофоустойчивость и автоматическое восстановление после сбоев
- Работа в одnodатацентровом и геораспределённом режимах, в том числе в Yandex Cloud в режиме Serverless

# Немного цифр о YDB Topics в Яндексе

**×200** Гбайт/с

80 + 120  
>300 на пике

**×21** МЛН  
EPS

6 + 15

**1500**

серверов

**5**

ДЦ

**30К**

ТОПИКОВ

**100К**

партиций

**9 млрд**

ИСТОЧНИКОВ

# Режимы записи

## Exactly once

- Режим по умолчанию
  - работает на большом количестве источников producer id
  - 10 000 (на партицию)
  - 9 000 000 000 (суммарно)
- Замедление обработки минимальное (поиск по хеш-таблице)

## At-least-once

Без дедупликации

# Запись в режиме Exactly-once

## Два поля в сообщениях

- Producer id
- Sequence number

Отправитель сам **назначает sequence number**, что позволяет переживать перезапуски отправителя и брокера

Аналог ключа идемпотентности, который обязателен



# Хранение **sequence number** в брокере

В брокере существует таблица метаданных, в том числе последний **sequence number** отправителей

Реализована атомарная запись самих данных и метаданных

При перезапуске отправителя восстанавливаются метаданные, в том числе **sequence number**

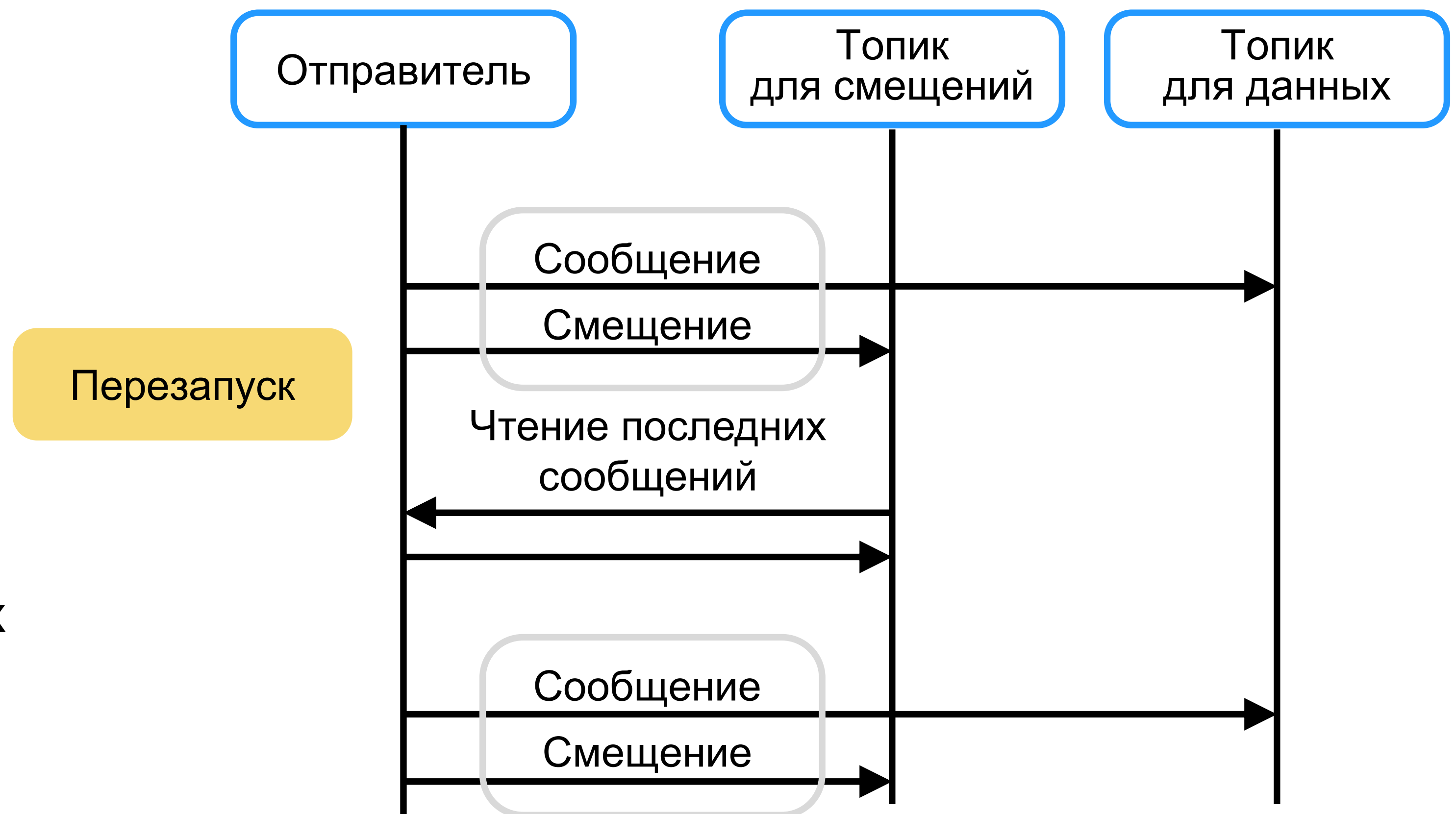
# Что делать отправителю при перезапуске

Ничего

Посмотреть записанное

## Хранить смещения

- Писать в транзакции в 2 топика
- Вторым топиком — сжатый, для смещений в топике данных
- Быстро прочитать последнее смещение



# Привязка отправителей к партициям

В брокере существует таблица  
«producer id, partition»

## Отправитель

- При перезапуске попадает в ту же партицию
- Нет нарушения порядка
- Переживает увеличение числа партиций без нарушения exactly-once

# Чтение в режиме Exactly-once

## Недостаток Transactional Inbox

Размер таблицы  
равен числу  
сообщений  
за большой  
промежуток времени

## Ситуация упрощается, когда

- есть гарантии порядка
- на записи уже реализована гарантия exactly-once

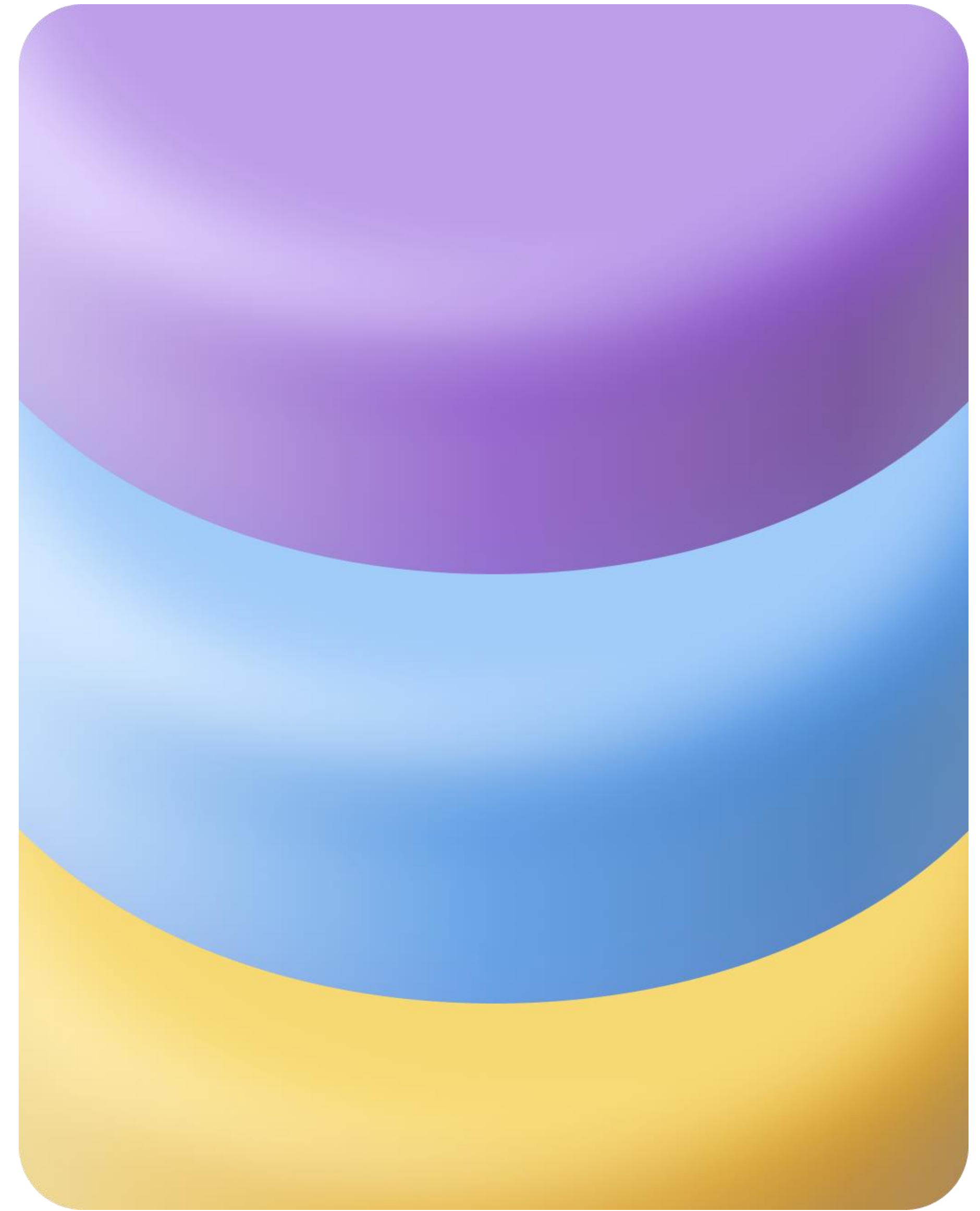
Теперь при чтении  
можно удалять  
дубли по паре

«Partition id, max  
offset» — разумное  
число 10 000

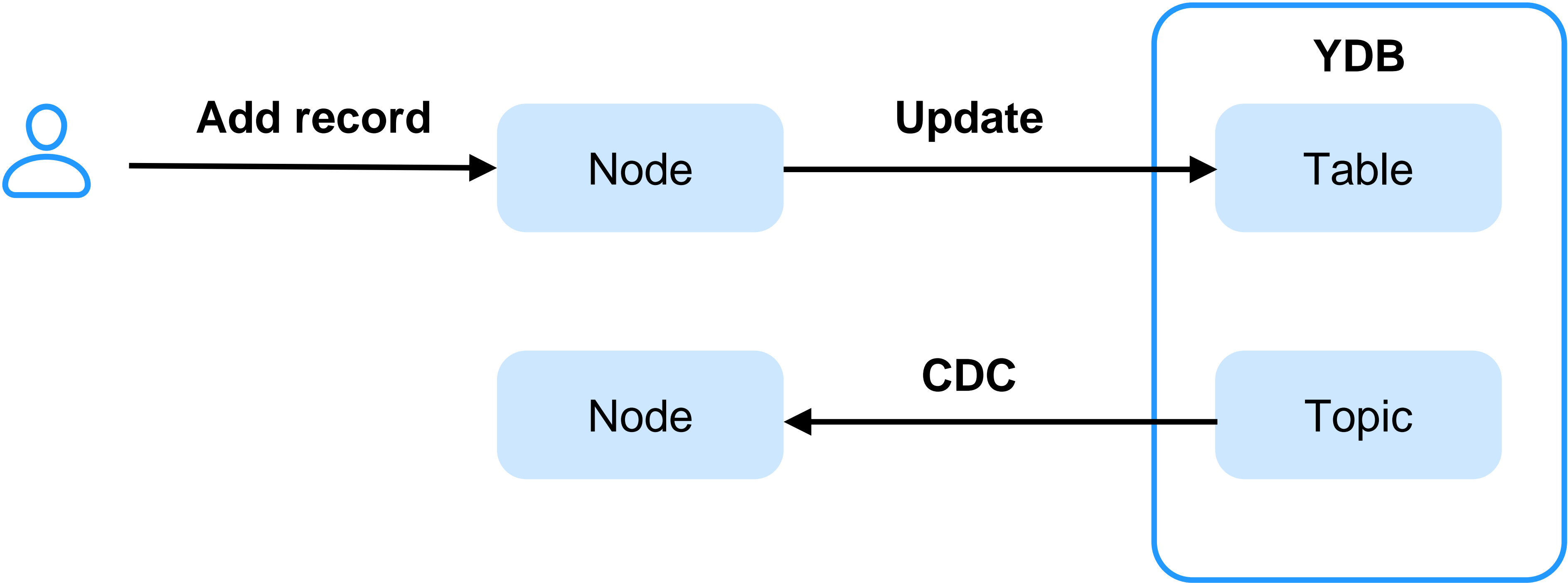
# Различия в гарантиях доставки Kafka и YDB Topics

	Kafka	YDB Topics
Переживает перезапуск сессии отправителя и брокера	Нет	Да
Порядковый номер доступен получателю	Нет	Да
Transactional outbox	Нужно	Не требуется
Transactional inbox	Нужно	Не требуется
Дедупликация на чтении	Храним много ключей идемпотентности	Одно максимальное смещение на партицию

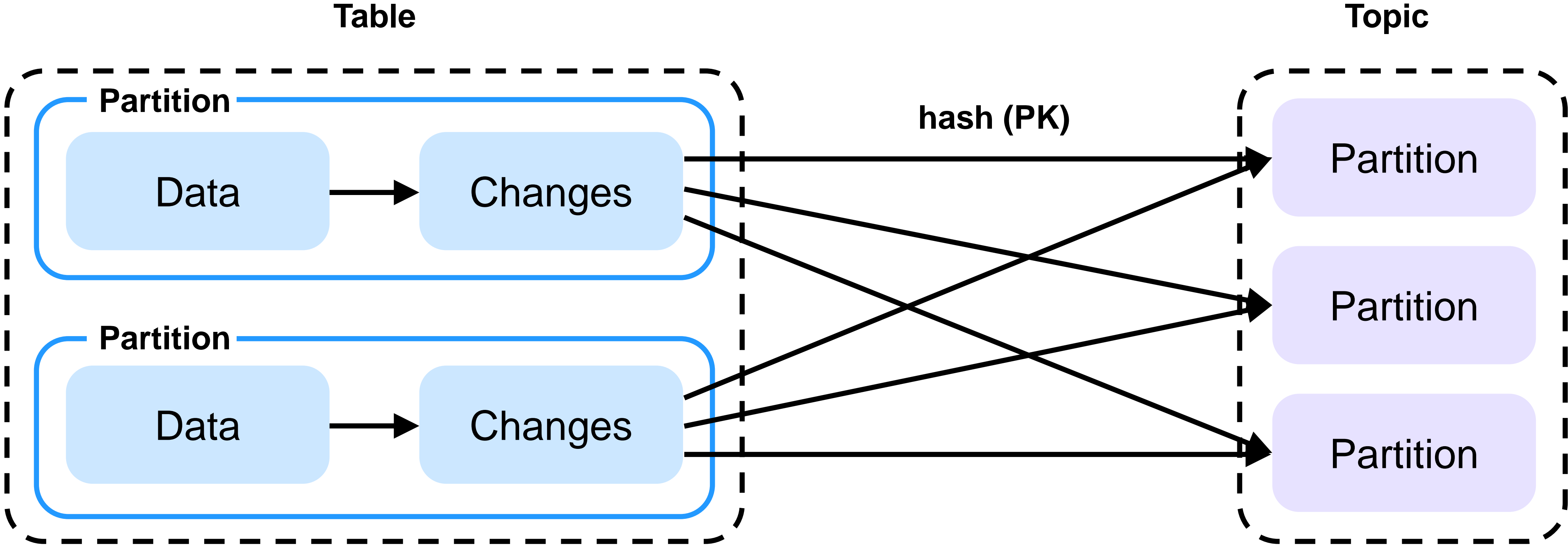
**YDV Topics:  
дополнение к  
гарантиям  
доставки**



# CDC в YDB: верхний уровень



# CDC в YDB: крупным планом





# CDC в YDB: свойства

- Все записи в очередь асинхронно
- Гарантии порядка
- Exactly-once
- TTL до 30 дней
- Увеличение потребления CPU на 1–10%
- Совместимость с Debezium





# Kafka API поверх YDB Topics

- Мы добавили Kafka API\*
- Узкие места в протоколе Kafka остались
- Можно бесшовно мигрировать с Kafka на YDB Topics
- Преимущества платформы YDB и повышенная производительность
- Получили Kafka в облаке в режиме `serverless/dedicated`
- Доступно в опенсорс

<https://ydb.tech/docs/ru/reference/kafka-api/>

\* версия 3.4.0.

# Совместимость Kafka API поверх YDB Topics

- ⌘ Kafka CLI
- ⌘ Kafka Connect
- ⌘ Kafka SDK
-  Logstash
-  Fluent Bit
- ...

# Итоги

- Потери/дубли — штатная ситуация
- Exactly-once — достижимо
- YDB Topics — расширенный протокол, Exactly-once из коробки
- YDB Topics — в ряде случаев At-least-once



Зевайкин Александр,  
Яндекс



**Saint  
HighLoad++**