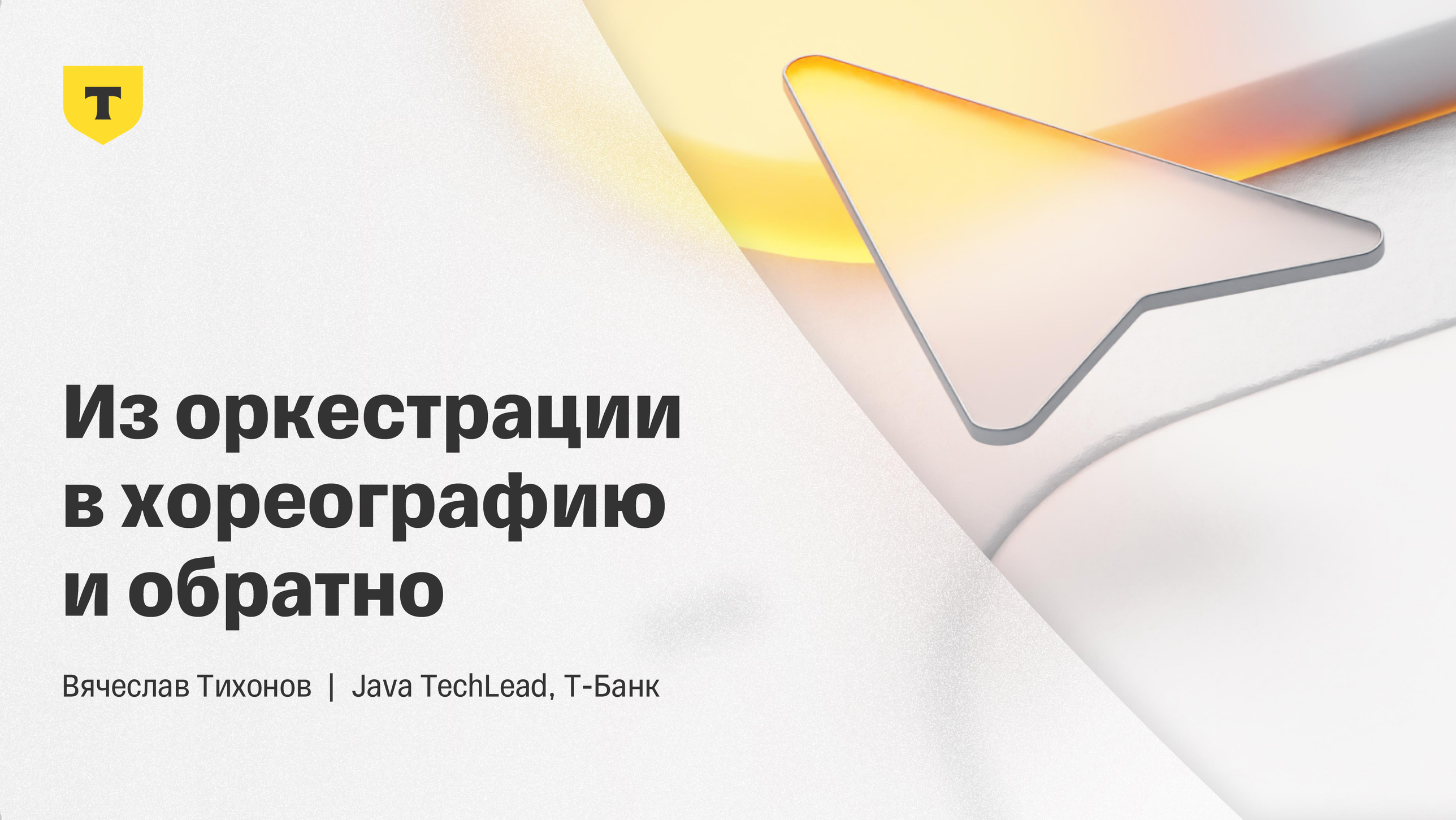


Из оркестрации в хореографию и обратно

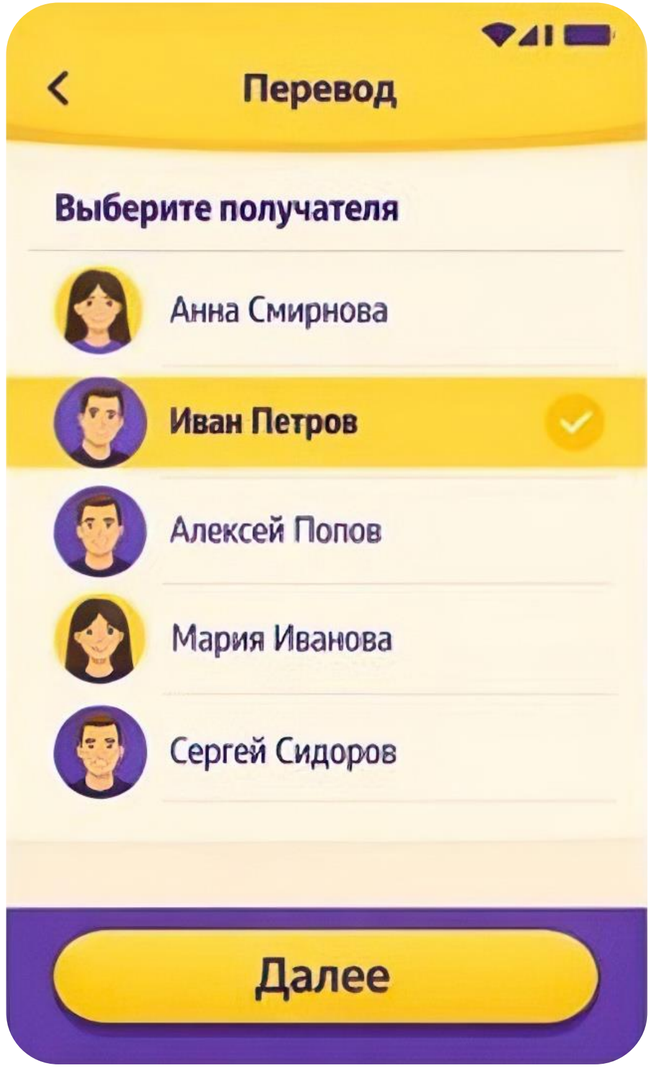
Вячеслав Тихонов | Java TechLead, Т-Банк

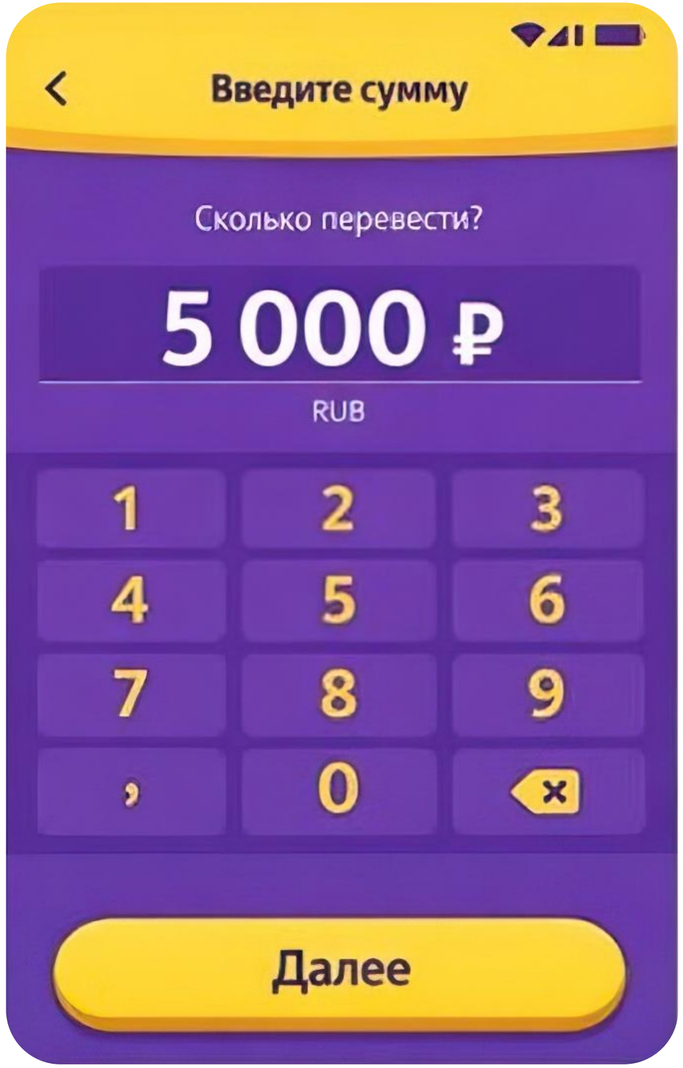
Вячеслав Тихонов

Java TechLead, Т-Банк

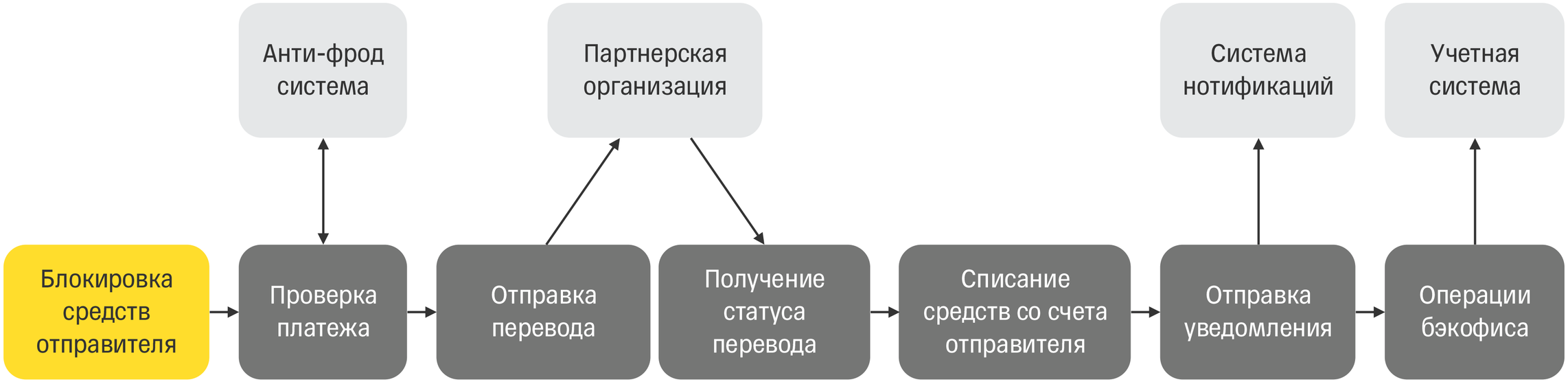
Хореография и оркестрация

Некоторые кнопки скрывают многое

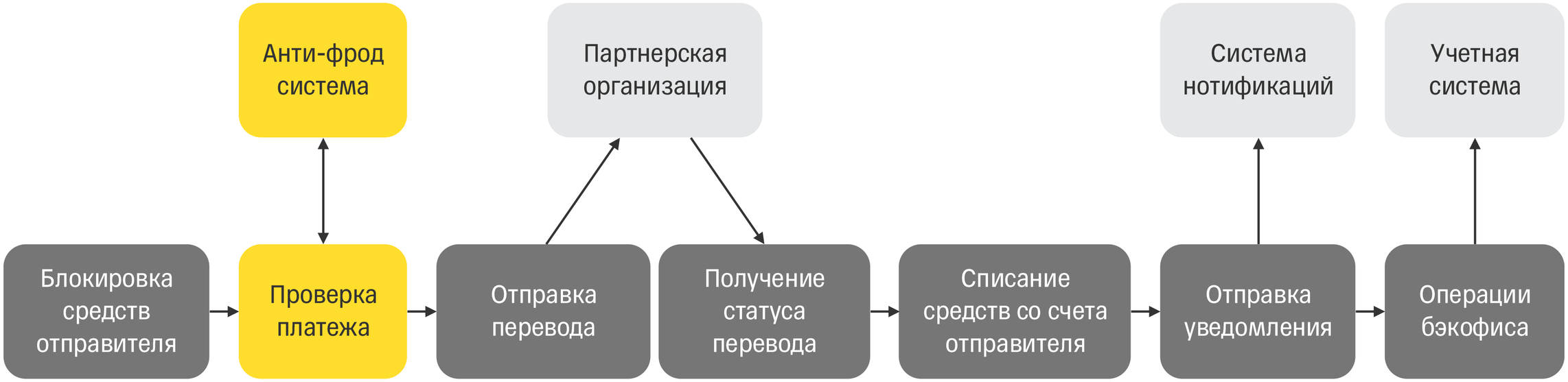




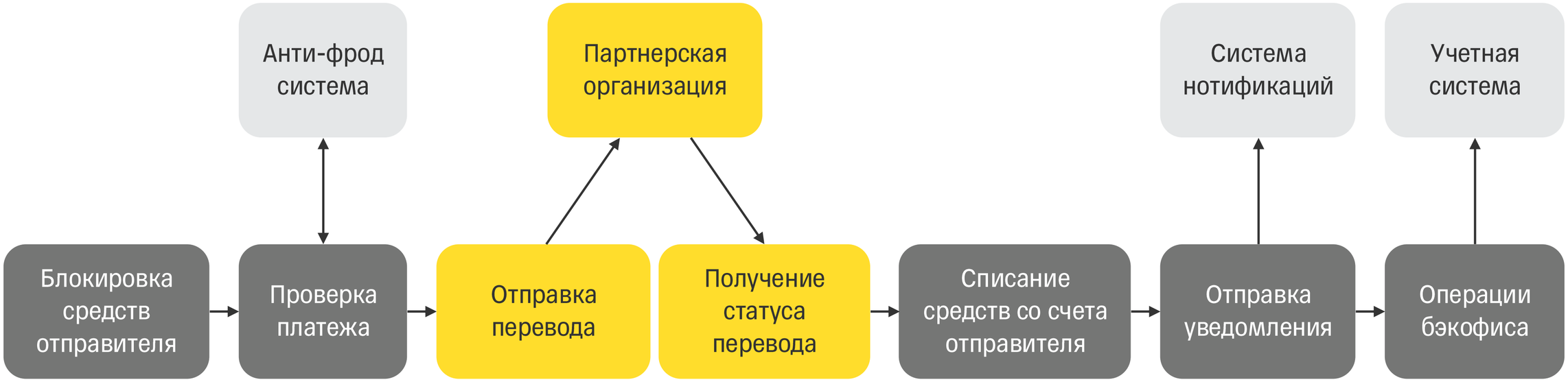
Финансовый перевод



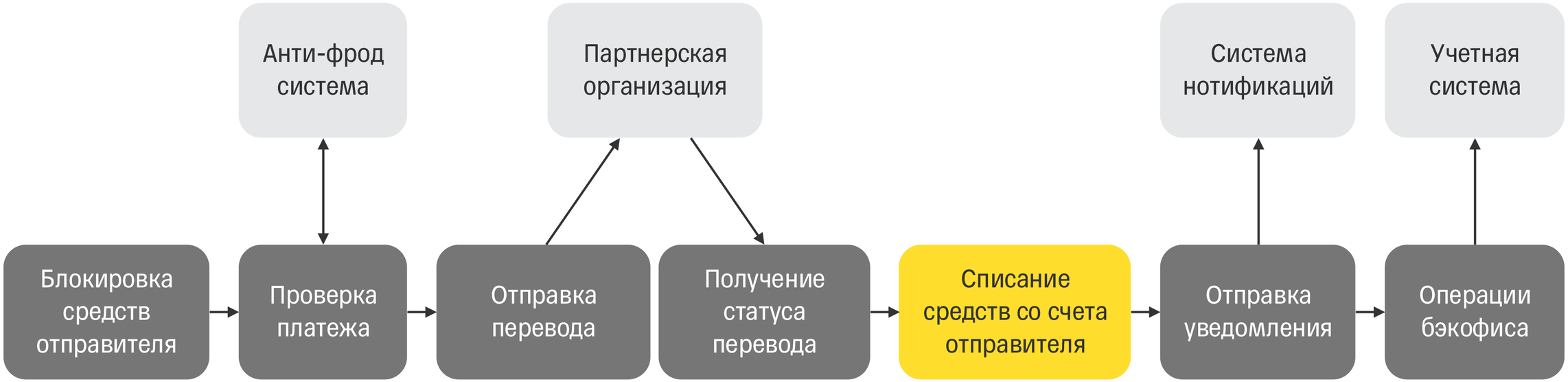
Финансовый перевод



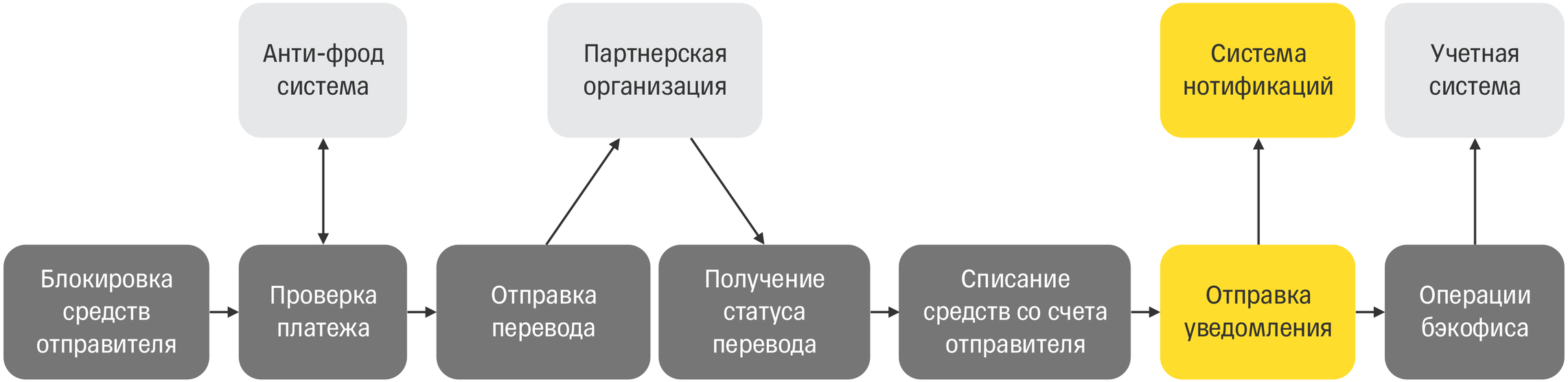
Финансовый перевод



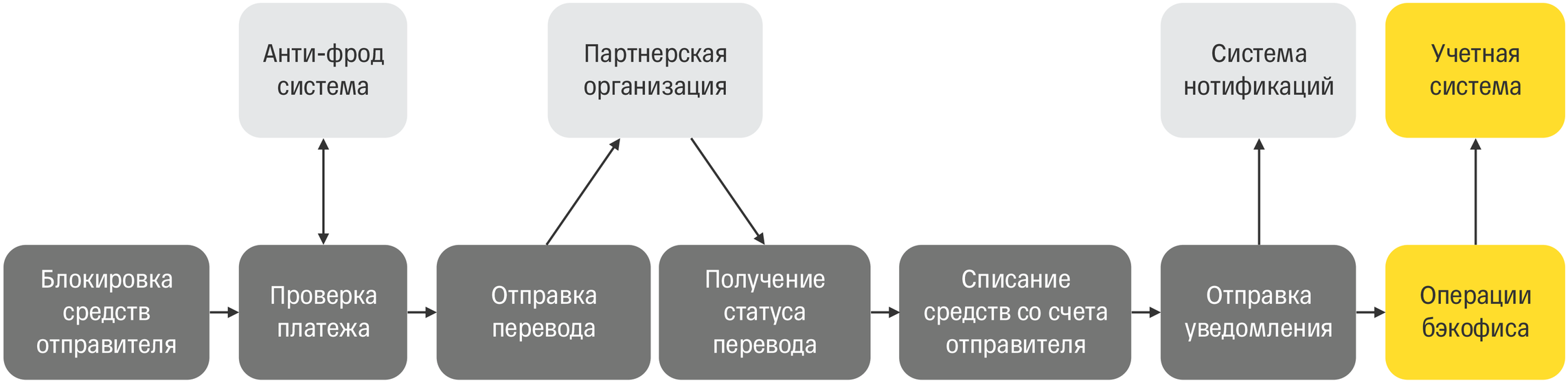
Финансовый перевод



Финансовый перевод



Финансовый перевод



Простая реализация

```
public Future<Void> moneyTransfer(MoneyTransferDto dto) {
    CompletableFuture.runAsync(() -> clientService.holdClientsAmount(dto))
            .thenRunAsync(() -> antiFraudService.validateExternalTransfer(dto))
            .thenApplyAsync(_ -> sbpIntegrationService.initiateSbpTransfer(dto))
            .thenAcceptAsync(paymentId ->
                    sbpIntegrationService.confirmSbpTransfer(paymentId))
            .thenRunAsync(() -> clientService.withdrawClientsAmount(dto))
            .thenRunAsync(() -> notificationService.sendNotification(
                    new NotificationDto(dto.clientId(), PUSH, SUCCESS_TRANSFER)))
            .thenRunAsync(() -> accountingService.accountExternalTransfer(dto));
}
```

Нефункциональные требования

Отказоустойчивость

Процесс не должен прерваться при падении частей системы

Нефункциональные требования

Отказоустойчивость

Процесс не должен прерваться при падении частей системы

Масштабируемость

Горизонтальная масштабируемость в соответствии с нагрузкой

Нефункциональные требования

Отказоустойчивость

Процесс не должен прерваться при падении частей системы

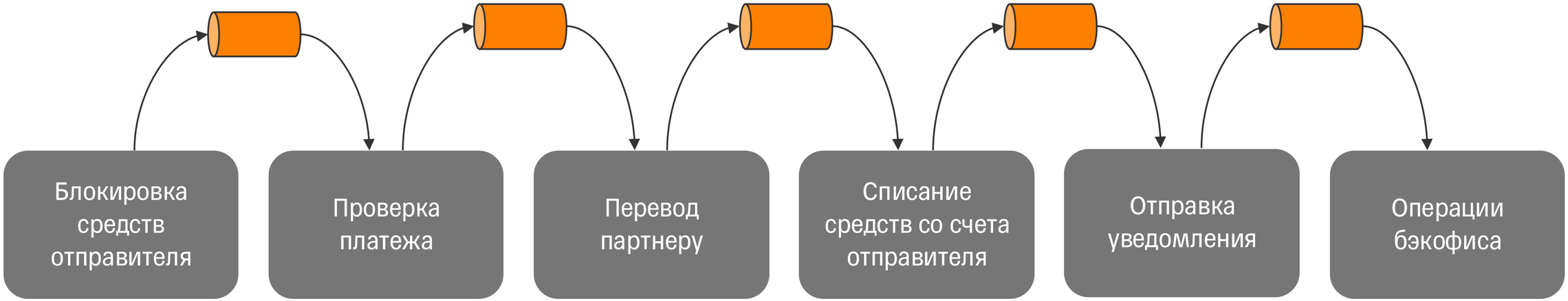
Масштабируемость

Горизонтальная масштабируемость в соответствии с нагрузкой

Целостность данных

При завершении процесса данные должны быть в консистентном состоянии

Взаимодействие через брокеры сообщений

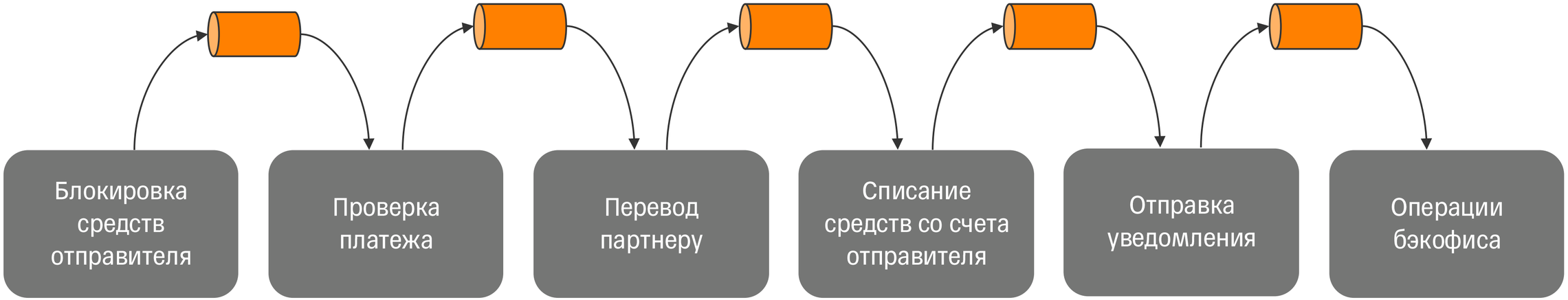


Реализация на Kafka

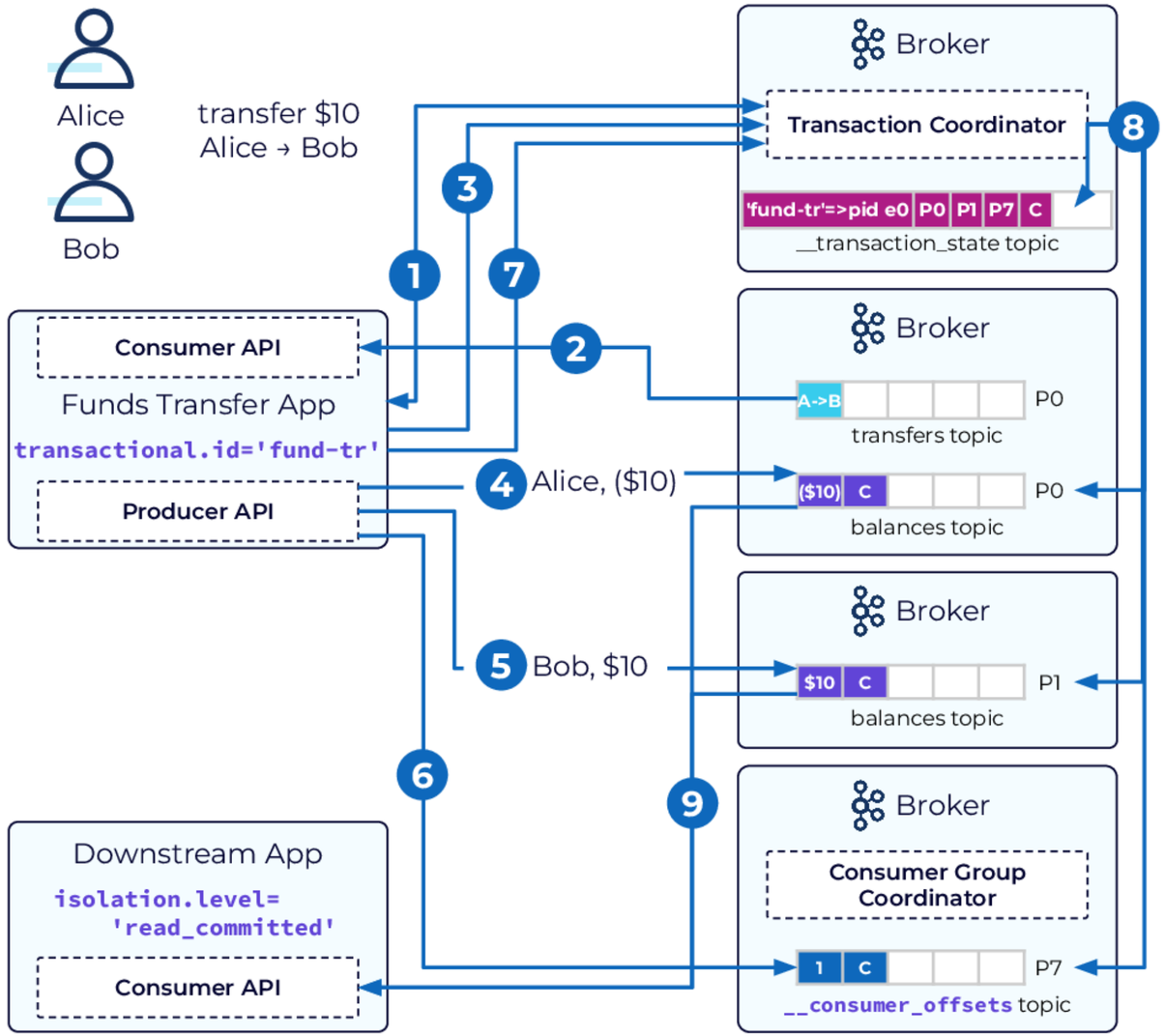
```java
@KafkaListener(topics = {"hold-topic"})
public void holdClientsAmount(MoneyTransferDto dto) {
    clientService.holdClientsAmount(dto);
    kafkaTemplate.send("anti-fraud-topic", dto);
}


@KafkaListener(topics = {"withdraw-topic"})
public void withdrawClientsAmount(MoneyTransferDto dto) {
    clientService.withdrawClientsAmount(dto);
    var notification = new NotificationDto(dto.clientId(),
                                           PUSH,
                                           SUCCESS_TRANSFER,
                                           Map.of("amount", dto.amount()));
    kafkaTemplate.send("notification-topic", notification);
}
```

Exactly once семантика



Kafka
Exactly once



https://developer.confluent.io/courses/architecture/transactions/

Kafka Exactly once

```
consumer:
  isolation-level: read_committed
  properties:
    enable.auto.commit: false
producer:
  transaction-id-prefix: ”app-tx-"
  acks: all
  properties:
    enable.idempotence: true
    transaction.timeout.ms: 300000
```

Kafka Exactly once

```
consumer:
  isolation-level: read_committed
  properties:
    enable.auto.commit: false
producer:
  transaction-id-prefix: ”app-tx-"
  acks: all
  properties:
    enable.idempotence: true
    max.in.flight.requests.per.connection: 5
    transaction.timeout.ms: 300000
```

Kafka Exactly once

```
@KafkaListener(topics = {"withdraw-topic"})
@Transactional("kafkaTransactionManager")
public void withdrawClientsAmount(MoneyTransferDto dto) {
    clientService.withdrawClientsAmount(dto);
    var notification = new NotificationDto(dto.clientId(),
                                           PUSH,
                                           SUCCESS_TRANSFER,
                                           Map.of("amount", dto.amount()));
    kafkaTemplate.send("notification-topic", notification);
}
```

Exactly once обработка

```
@KafkaListener(topics = {"withdraw-topic"})
@Transactional("kafkaTransactionManager")
public void withdrawClientsAmount(MoneyTransferDto dto) {
    if (clientService.isNewWithdraw(dto)) {
        clientService.withdrawClientsAmount(dto);
        var notification = new NotificationDto(dto.clientId(),
                                               PUSH,
                                               SUCCESS_TRANSFER,
                                               Map.of("amount", dto.amount()));
        kafkaTemplate.send("notification-topic", notification);
    }
}
```

Exactly once обработка

```
@KafkaListener(topics = {"withdraw-topic"})
@Transactional("kafkaTransactionManager")
public void withdrawClientsAmount(MoneyTransferDto dto) {
    if (clientService.isNewWithdraw(dto)) {
        clientService.withdrawClientsAmount(dto);
        var notification = new NotificationDto(dto.clientId(),
                                               PUSH,
                                               SUCCESS_TRANSFER,
                                               Map.of("amount", dto.amount()));
        kafkaTemplate.send("notification-topic", notification);
    }
}
```

Saga


Блокировка средств отправителя
Проверка платежа
Перевод партнеру
Списание средств со счета отправителя
Отправка уведомления
Операции бэкофиса
Разблокировка средств
Отмена перевода
Корректировка для возврата средств

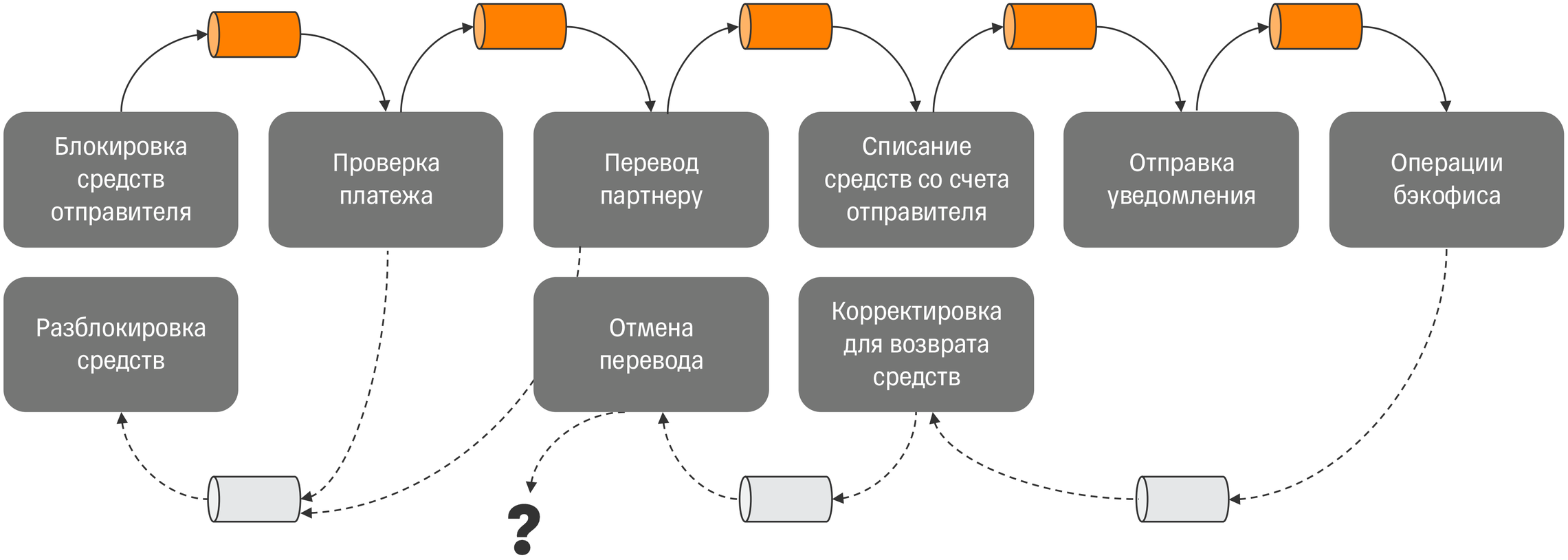
Бизнесовые ошибки

```
@KafkaListener(topics = {"payment-validation-topic"})
public void validateMoneyTransfer(MoneyTransferDto dto) {
    AntiFraudResult validationResult = antiFraudService.validateExternalTransfer(dto);
    switch (validationResult) {
        case ALLOWED -> {
            kafkaTemplate.send("payment-gate-topic", dto);
        }
        case SUSPICIOUS, BLOCKED -> {
            kafkaTemplate.send("unhold-topic", dto);
        }
    }
}
```

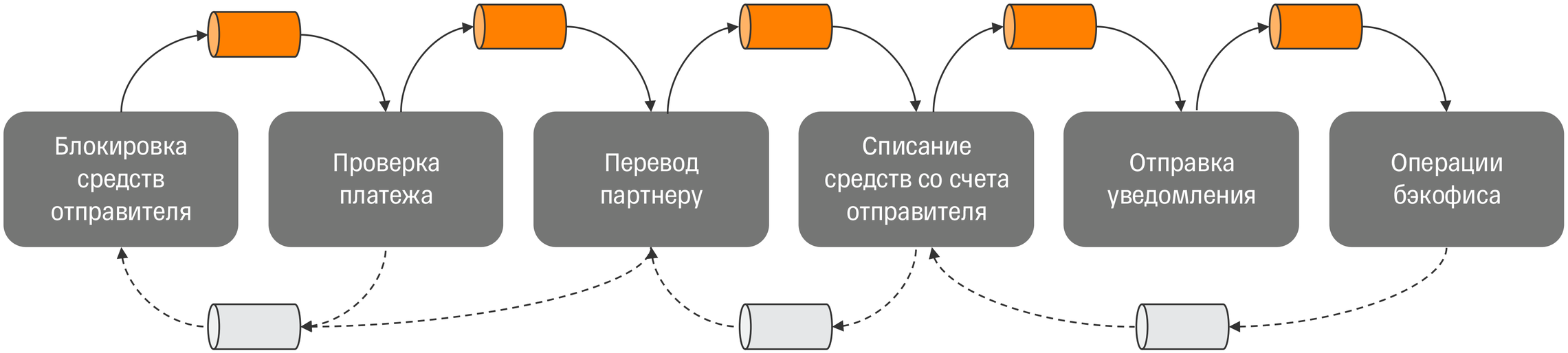
Обработка исключений

```
@KafkaListener(topics = {"payment-validation-topic"},
               errorHandler = "paymentValidationErrorHandler")
public void validateMoneyTransfer(MoneyTransferDto dto) {
    AntiFraudResult validationResult = antiFraudService.validateExternalTransfer(dto);
    switch (validationResult) {
        case ALLOWED -> {
            kafkaTemplate.send("payment-gate-topic", dto);
        }
        case SUSPICIOUS, BLOCKED -> {
            kafkaTemplate.send("unhold-topic", dto);
        }
    }
}
```

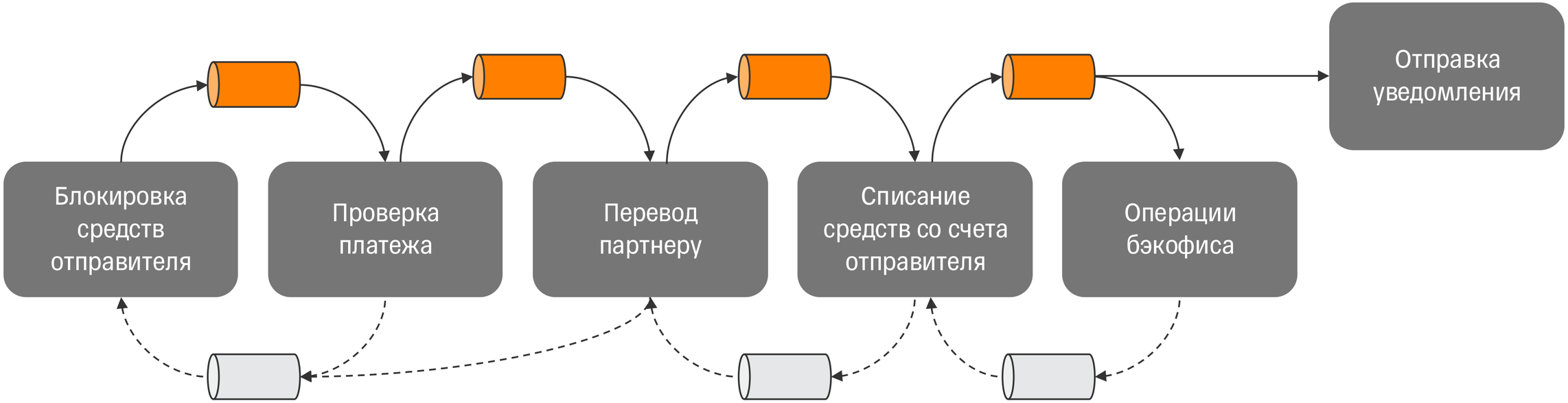
Корректировка на ошибки в корректировках



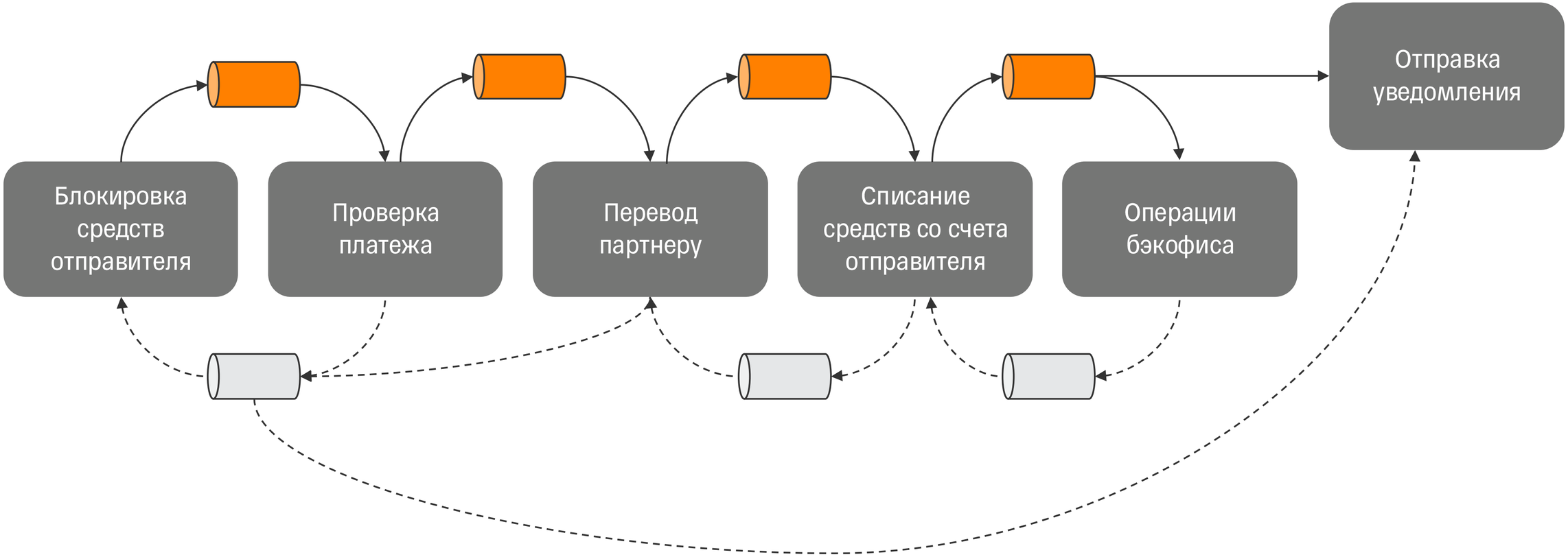
Слабая связность?



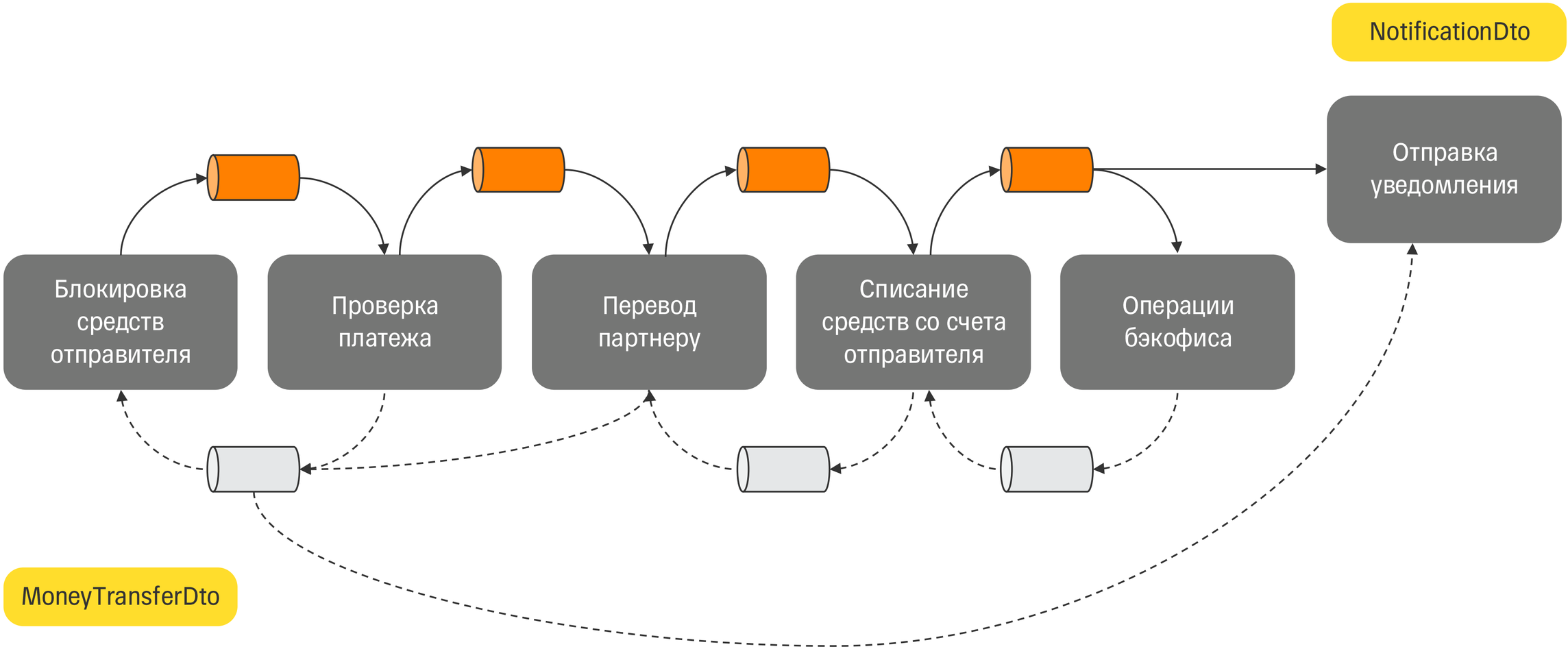
Слабая связность?



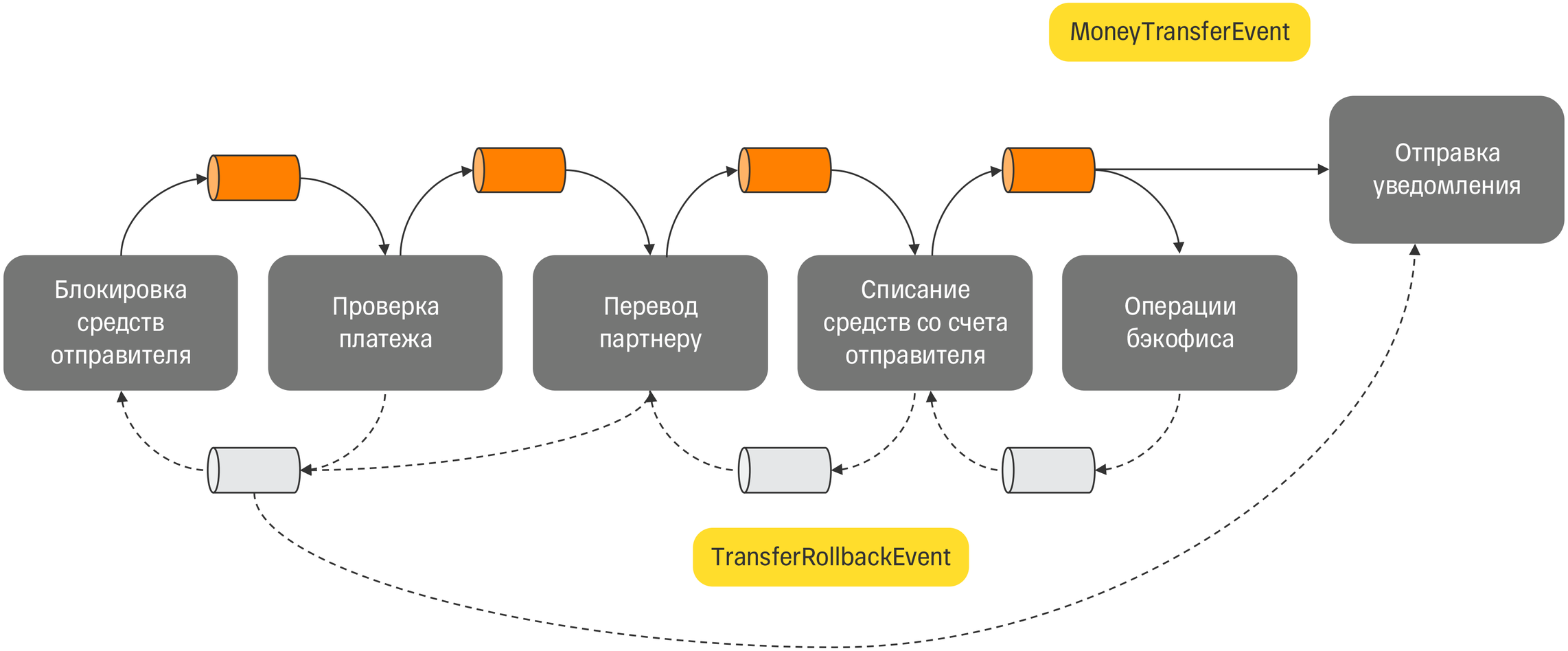
Слабая связность?



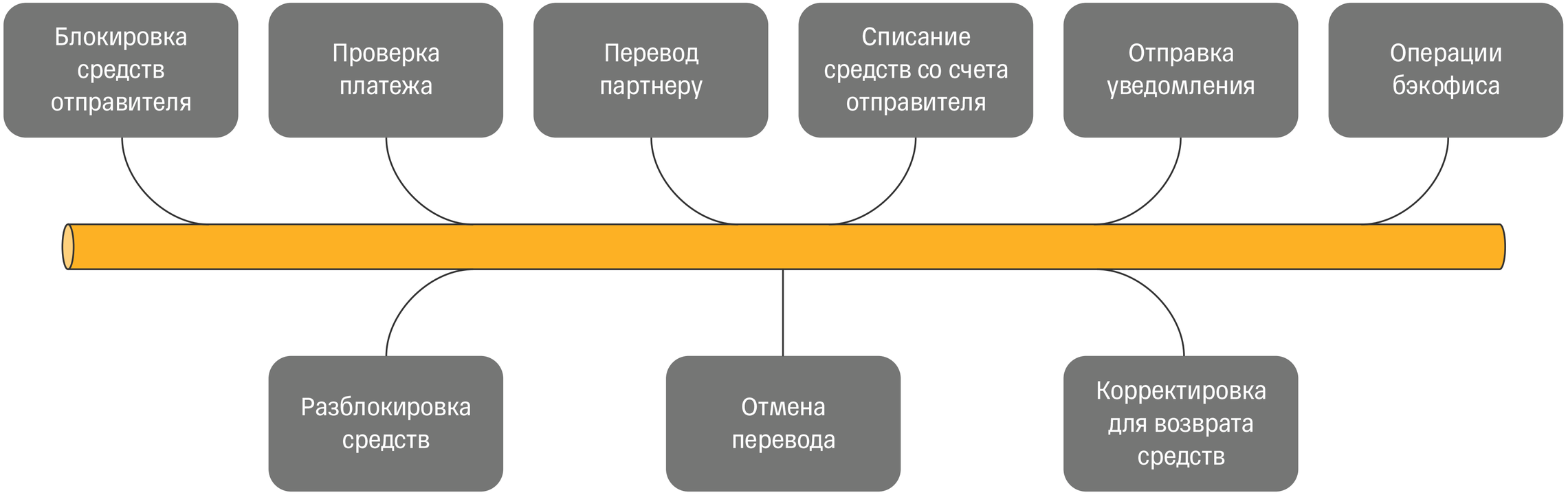
Слабая связность?



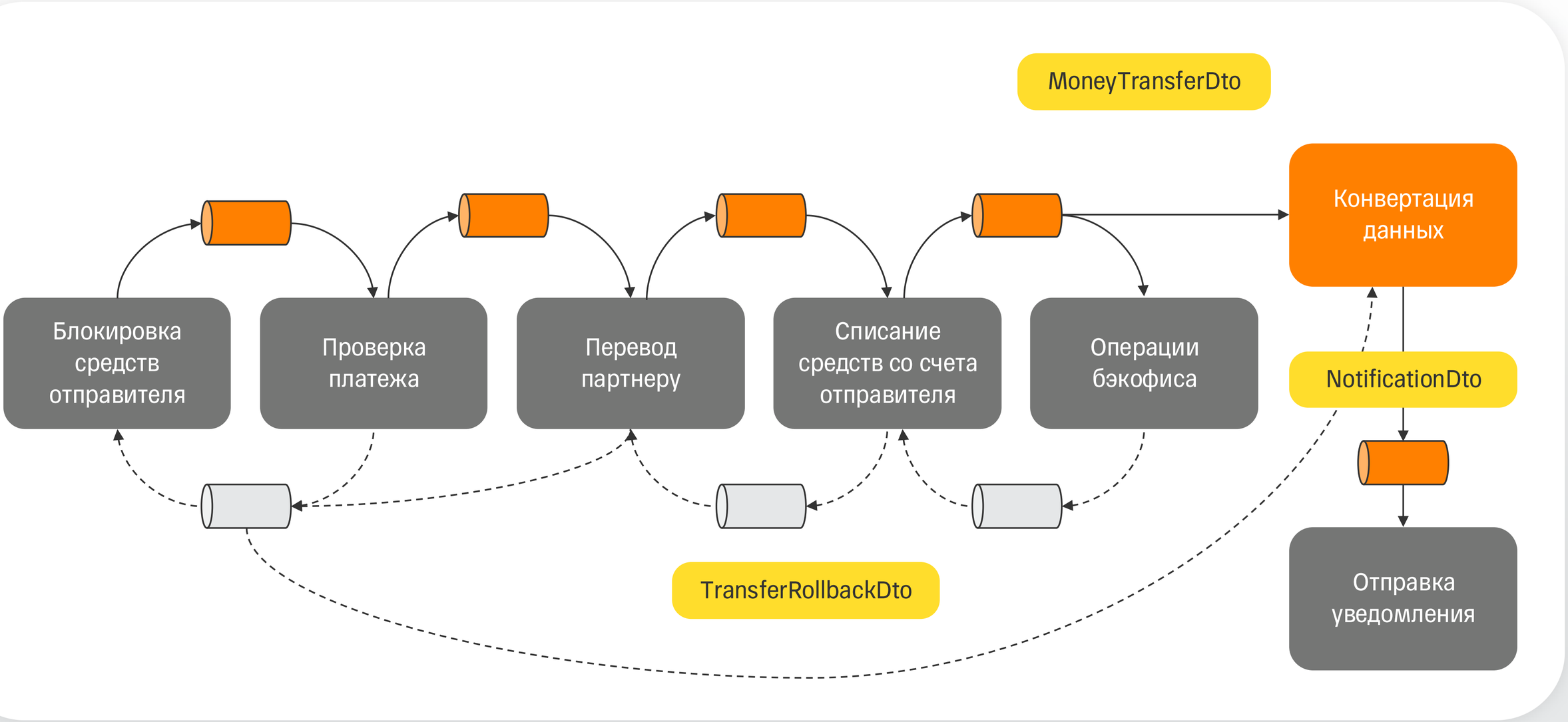
Событийная модель



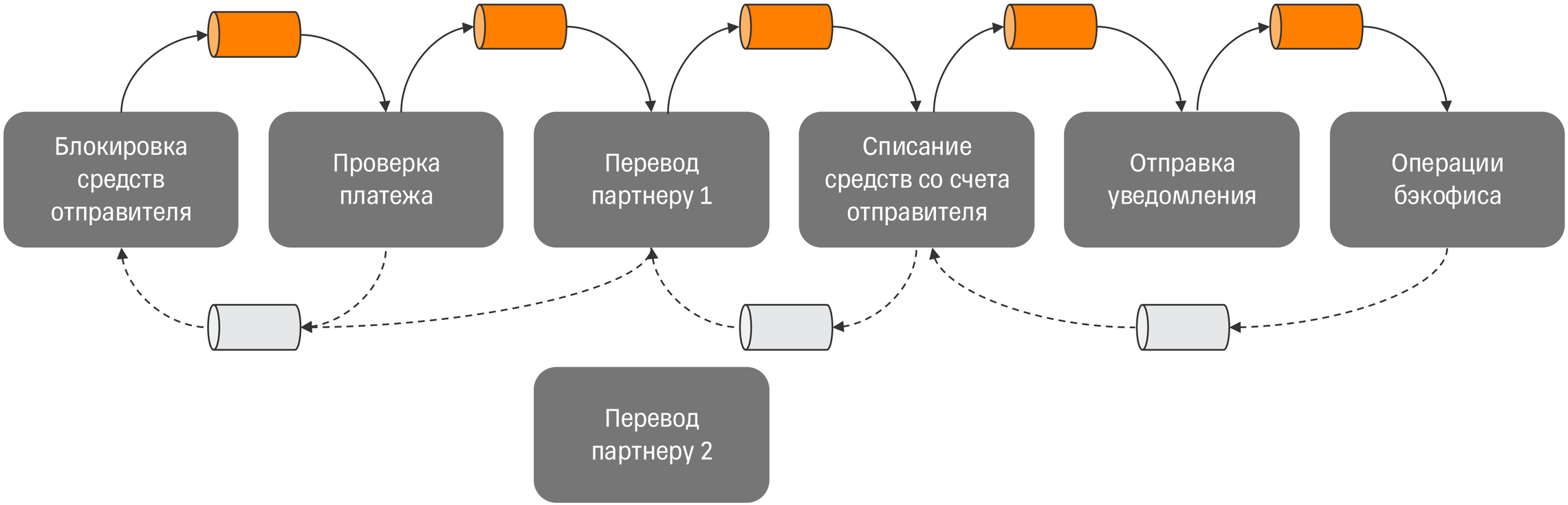
Шина данных


Блокировка средств отправителя
Проверка платежа
Перевод партнеру
Списание средств со счета отправителя
Отправка уведомления
Операции бэкофиса
Разблокировка средств
Отмена перевода
Корректировка для возврата средств

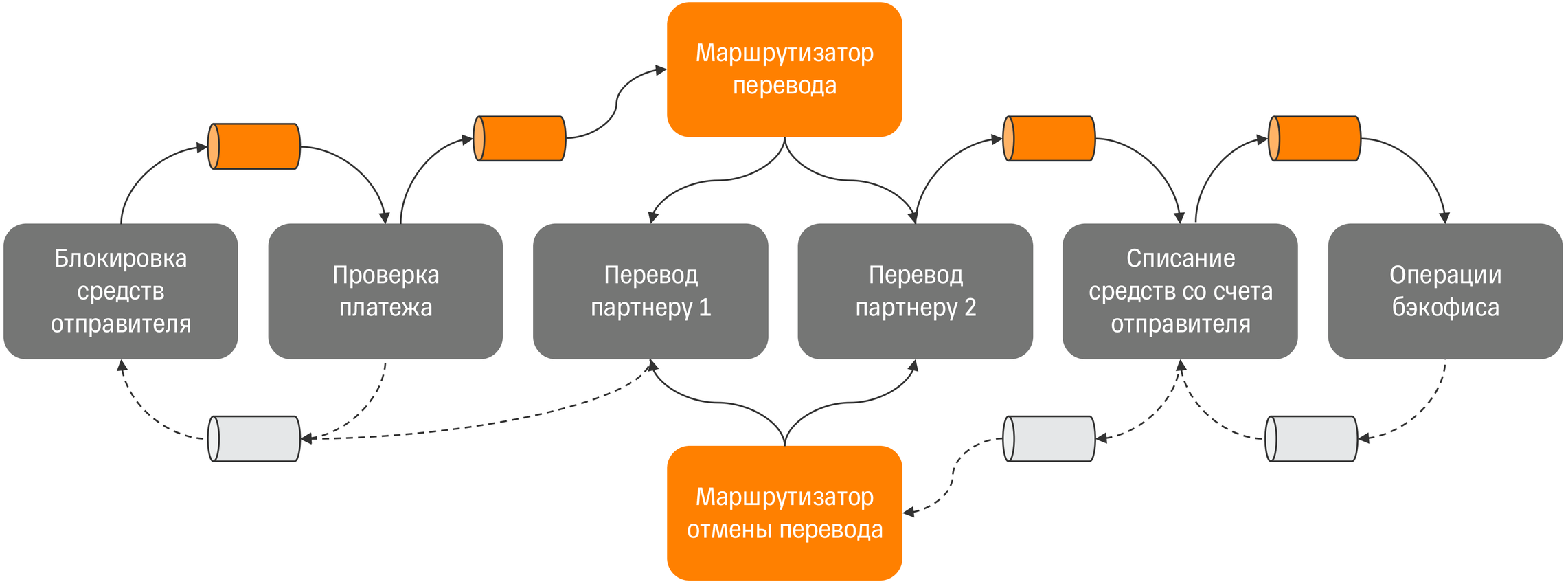
Дополнительный адаптер



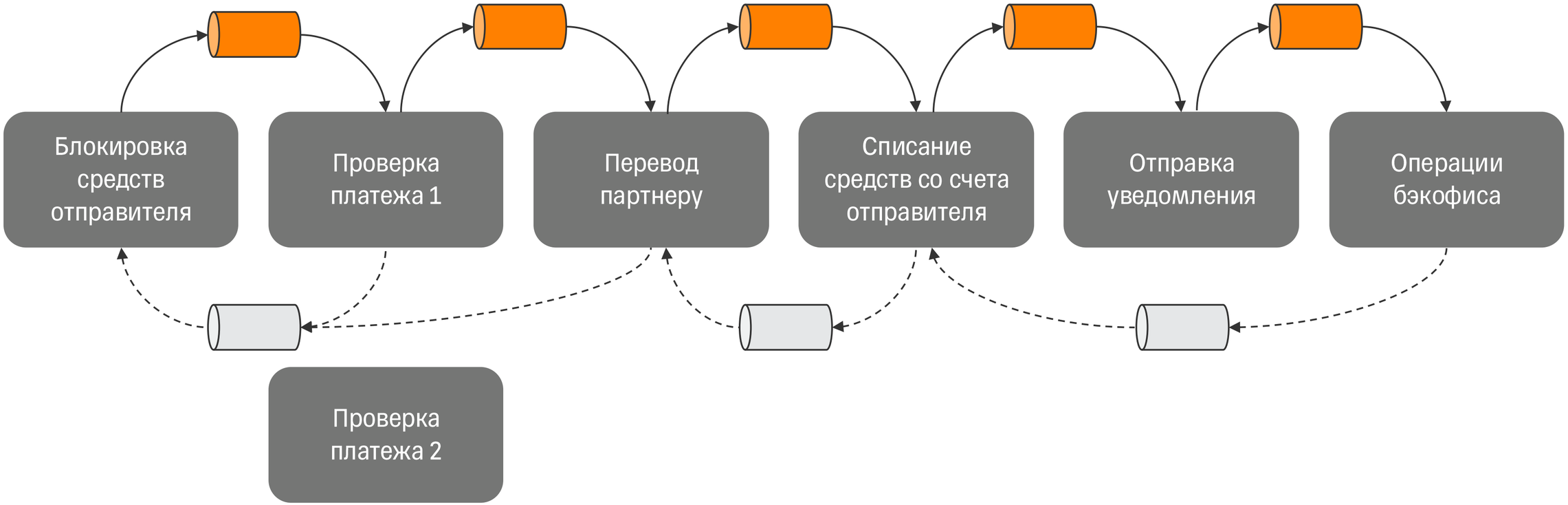
Ветвление процесса



Дополнительный маршрутизатор

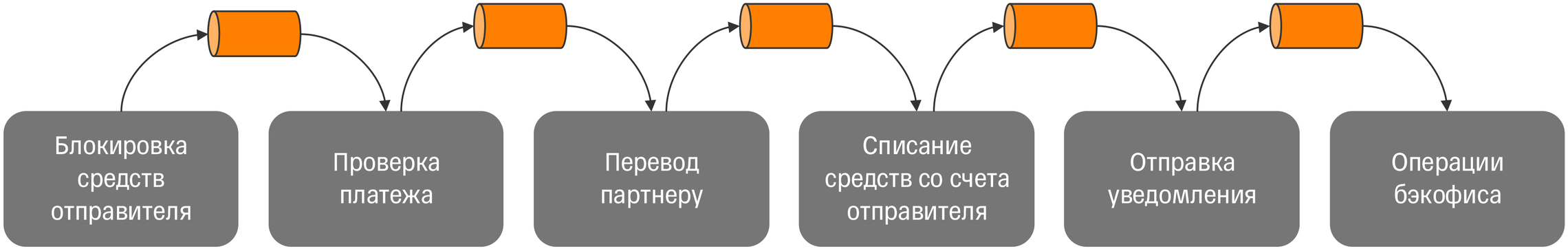


Параллельное выполнение



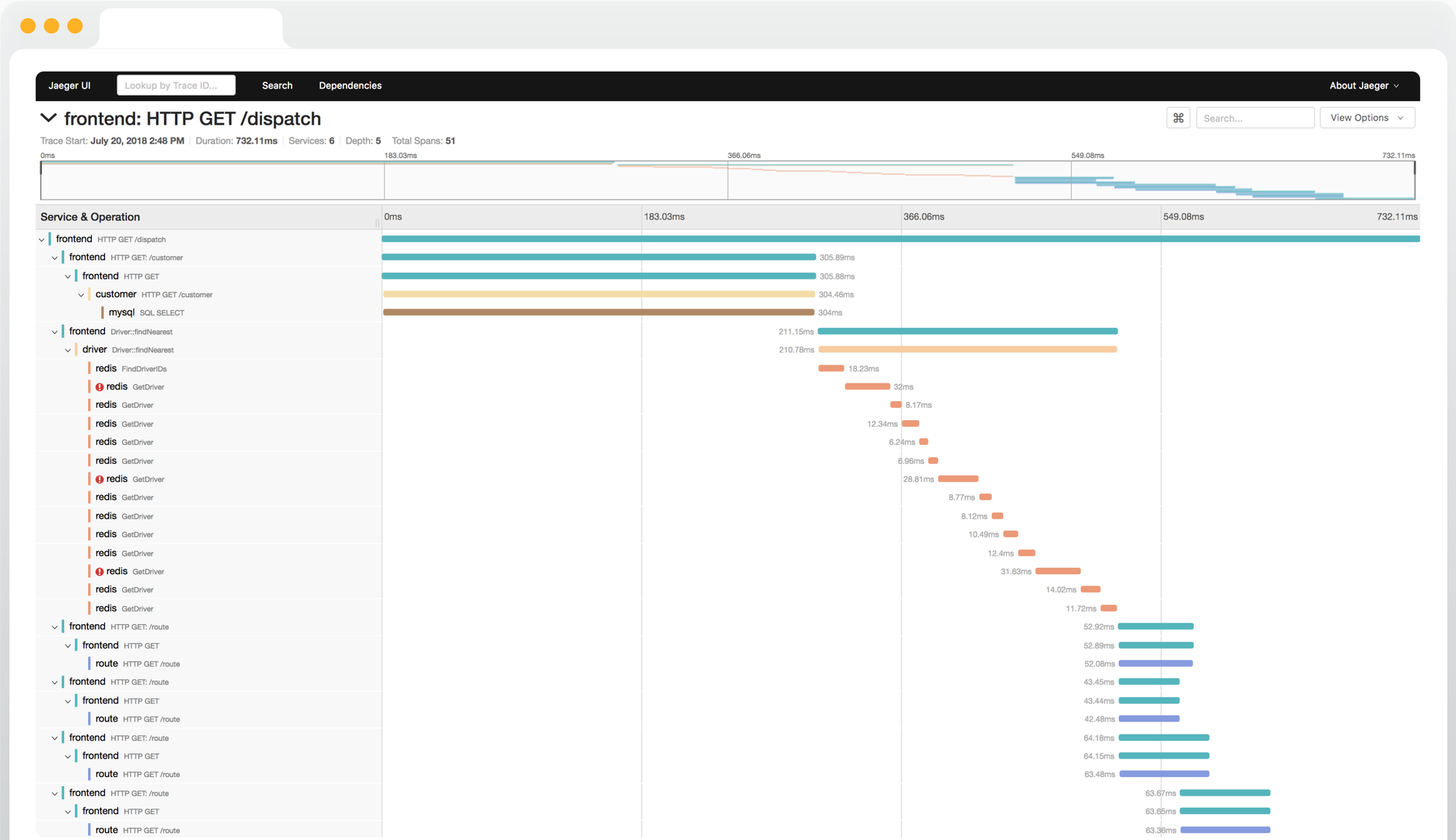
Дополнительный агрегатор

Изначальный дизайн

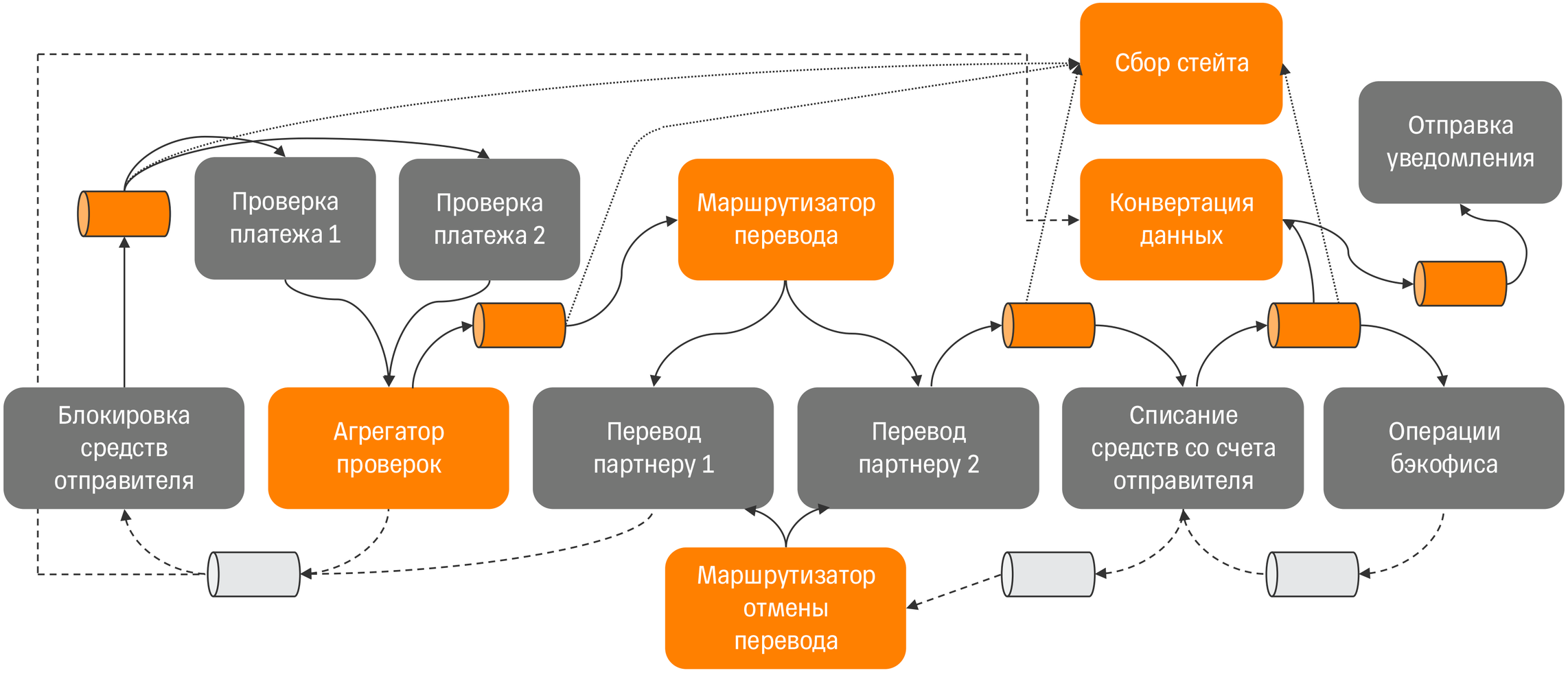


Через несколько спринтов

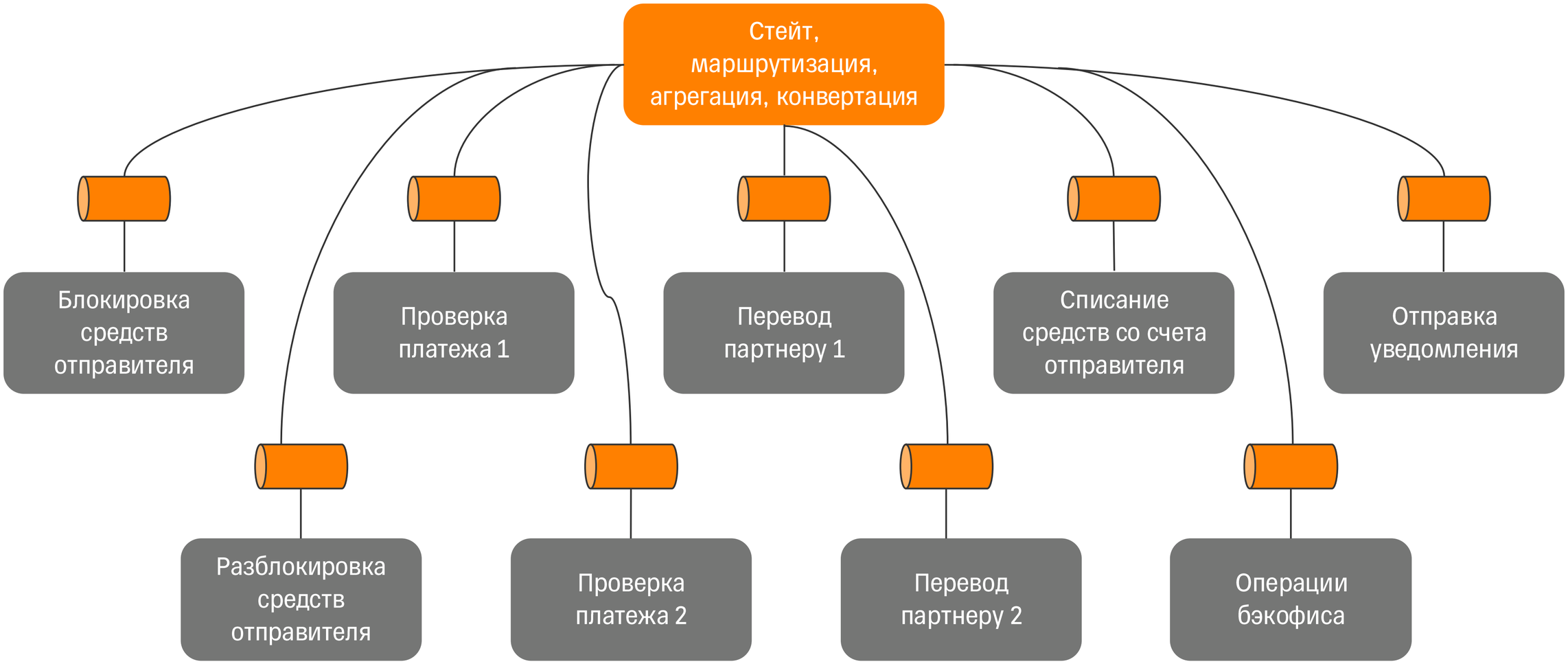
Мониторинг


Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
frontend: HTTP GET /dispatch
Search...
View Options
Trace Start: July 20, 2018 2:48 PM | Duration: 732.11ms | Services: 6 | Depth: 5 | Total Spans: 51
0ms
183.03ms
366.06ms
549.08ms
732.11ms
Service & Operation
frontend HTTP GET /dispatch
frontend HTTP GET: /customer
305.89ms
frontend HTTP GET
305.88ms
customer HTTP GET /customer
304.46ms
mysql SQL SELECT
304ms
frontend Driver::findNearest
211.15ms
driver Driver::findNearest
210.78ms
redis FindDriverIDs
18.23ms
redis GetDriver
32ms
redis GetDriver
8.17ms
redis GetDriver
12.34ms
redis GetDriver
6.24ms
redis GetDriver
6.96ms
redis GetDriver
28.81ms
redis GetDriver
8.77ms
redis GetDriver
8.12ms
redis GetDriver
10.49ms
redis GetDriver
12.4ms
redis GetDriver
31.63ms
redis GetDriver
14.02ms
redis GetDriver
11.72ms
frontend HTTP GET: /route
52.92ms
frontend HTTP GET
52.89ms
route HTTP GET /route
52.08ms
frontend HTTP GET: /route
43.45ms
frontend HTTP GET
43.44ms
route HTTP GET /route
42.48ms
frontend HTTP GET: /route
64.18ms
frontend HTTP GET
64.15ms
route HTTP GET /route
63.48ms
frontend HTTP GET: /route
63.67ms
frontend HTTP GET
63.65ms
route HTTP GET /route
63.36ms

Стейт процесса через event sourcing



А не собрать ли все в одном месте?



Хореография в итоге

Exactly once

Требуется аккуратная
настройка как Kafka клиента,
так и TransactionManager в Spring.

Детерминизм шагов

Хореография в итоге

Exactly once

Требуется аккуратная
настройка как Kafka клиента,
так и TransactionManager в Spring.

Детерминизм шагов

Обработка ошибок

Ретраи и дополнительная обработка –
средствами Spring. Для ручной
обработки – только DLQ паттерн

Хореография в итоге

Exactly once

Требуется аккуратная настройка как Kafka клиента, так и TransactionManager в Spring.

Детерминизм шагов

Обработка ошибок

Ретраи и дополнительная обработка – средствами Spring. Для ручной обработки – только DLQ паттерн

Связность шагов

Связаны на уровне контрактов данных. Либо универсальный формат сообщений, либо дополнительный адаптер-слой

Хореография в итоге

Exactly once

Требуется аккуратная настройка как Kafka клиента, так и TransactionManager в Spring.

Детерминизм шагов

Обработка ошибок

Ретраи и дополнительная обработка – средствами Spring. Для ручной обработки – только DLQ паттерн

Связность шагов

Связаны на уровне контрактов данных. Либо универсальный формат сообщений, либо дополнительный адаптер-слой

Ветвления и агрегация

Требуется дополнительное архитектурное решение

Хореография в итоге

Exactly once

Требуется аккуратная настройка как Kafka клиента, так и TransactionManager в Spring.

Детерминизм шагов

Обработка ошибок

Ретраи и дополнительная обработка – средствами Spring. Для ручной обработки – только DLQ паттерн

Связность шагов

Связаны на уровне контрактов данных. Либо универсальный формат сообщений, либо дополнительный адаптер-слой

Ветвления и агрегация

Требуется дополнительное архитектурное решение

Мониторинг процессов

Трейсинг и логи

Хореография в итоге

Exactly once

Требуется аккуратная настройка как Kafka клиента, так и TransactionManager в Spring.

Детерминизм шагов

Обработка ошибок

Ретраи и дополнительная обработка – средствами Spring. Для ручной обработки – только DLQ паттерн

Связность шагов

Связаны на уровне контрактов данных. Либо универсальный формат сообщений, либо дополнительный адаптер-слой

Ветвления и агрегация

Требуется дополнительное архитектурное решение

Мониторинг процессов

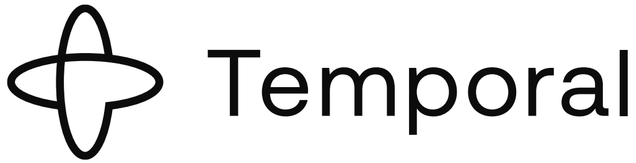
Трейсинг и логи

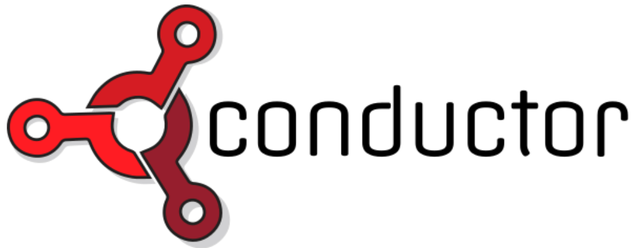
Управление процессами

Скрипты или дополнительное решение

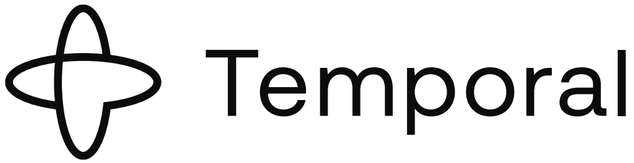
Может есть готовое решение?

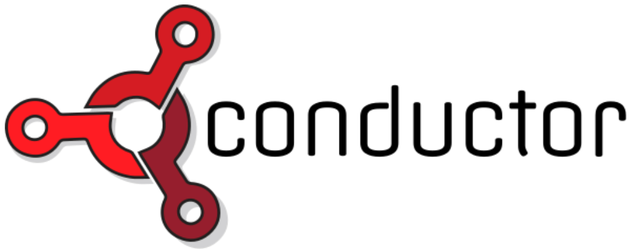
Может есть готовое решение?





Может есть готовое решение?





Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Обработка ошибок

Средствами SDK

Инструменты UI

Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Обработка ошибок

Средствами SDK

Инструменты UI

Связность шагов

Связь только с оркестратором

Трансформация данных доступна
на стороне оркестратора

Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Обработка ошибок

Средствами SDK

Инструменты UI

Связность шагов

Связь только с оркестратором

Трансформация данных доступна
на стороне оркестратора

Ветвления и агрегация

Возможность декларативного
или императивного описания
процесса

Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Обработка ошибок

Средствами SDK

Инструменты UI

Связность шагов

Связь только с оркестратором

Трансформация данных доступна
на стороне оркестратора

Ветвления и агрегация

Возможность декларативного
или императивного описания
процесса

Мониторинг процессов

Трейсинг и логи

Агрегированное состояние

Инструменты UI

Готовый оркестратор

Exactly once

То же, что и в хореографии +
доп. возможности благодаря стейту

Детерминизм шагов

Обработка ошибок

Средствами SDK

Инструменты UI

Связность шагов

Связь только с оркестратором

Трансформация данных доступна
на стороне оркестратора

Ветвления и агрегация

Возможность декларативного
или императивного описания
процесса

Мониторинг процессов

Трейсинг и логи

Агрегированное состояние

Инструменты UI

Управление процессами

Инструменты SDK

Инструменты UI

Пример workflow Temporal

```
@Override
public void moneyTransfer(MoneyTransferDto dto) {
     clientActivity.holdClientsAmount(dto);
     antiFraudActivity.validateExternalTransfer(dto);
     var paymentId = sbpIntegrationAcivity.initiateSbpTransfer(dto);
     sbpIntegrationActivity.confirmSbpTransfer(paymentId);
     clientActivity.withdrawClientsAmount(dto);
     var notification = new NotificationDto(dto.clientId(), PUSH, SUCCESS_TRANSFER);
     notificationActivity.sendNotification(notification);
     accountingActivity.accountExternalTransfer(dto);
}
```

Недостатки оркестрации

Единая точка отказа

Такой же инфраструктурный компонент,
как и Kafka, база данных т.д.

Недостатки оркестрации

Единая точка отказа

Такой же инфраструктурный компонент,
как и Kafka, база данных т.д.

Сложность развертывания

Большинство оркестраторов предоставляют
Docker Compose файлы и Helm чарты

Недостатки оркестрации

Единая точка отказа

Такой же инфраструктурный компонент,
как и Kafka, база данных т.д.

Сложность развертывания

Большинство оркестраторов предоставляют
Docker Compose файлы и Helm чарты

Узкое место системы

Производительность оркестратора зависит
от выбранного решения

Недостатки оркестрации

Единая точка отказа

Такой же инфраструктурный компонент,
как и Kafka, база данных т.д.

Сложность развертывания

Большинство оркестраторов предоставляют
Docker Compose файлы и Helm чарты

Узкое место системы

Производительность оркестратора зависит
от выбранного решения

Координация команд

Если нет договоренностей между командами
об использовании оркестратора, придется делать
дополнительные адаптеры

Вопросы?

No-code
Python
JavaScript
Node.js
Swift
Mobile App
Development
Innovation
Golang
Java
Analysis
Planning