

Здравствуйте, меня зовут Дмитрий Карловский и я.. не доверяю своему серверу. Мне кажется, что *админ читает* мои личные сообщения, а потом рассылает *от моего имени* дикпики прекрасным дамам. Так что если в очередном *стиве БД* вы обнаружите мои стрёмные фотки, то знайте, что это не я. Честное пионерское! А вот если обнаружите красивые, то обращайтесь, могу прислать ещё.

Давайте подумаем, как решить эти проблемы, и создадим совершенно новую архитектуру, в корне решающую не только их, но и многие другие. Итак, цепляйтесь за спойлеры и поехали: *local-first, real-time, confict-free, optimistic-ui, de-centralized, zero-trust, graph-db, crypto-auth.*





 Дмитрий Карловский

30 лет в IT, но всё ещё не скучился

 Яндекс,  1С,  Газпром

 Wrike,  SAPRUN,  Deutsche Bank

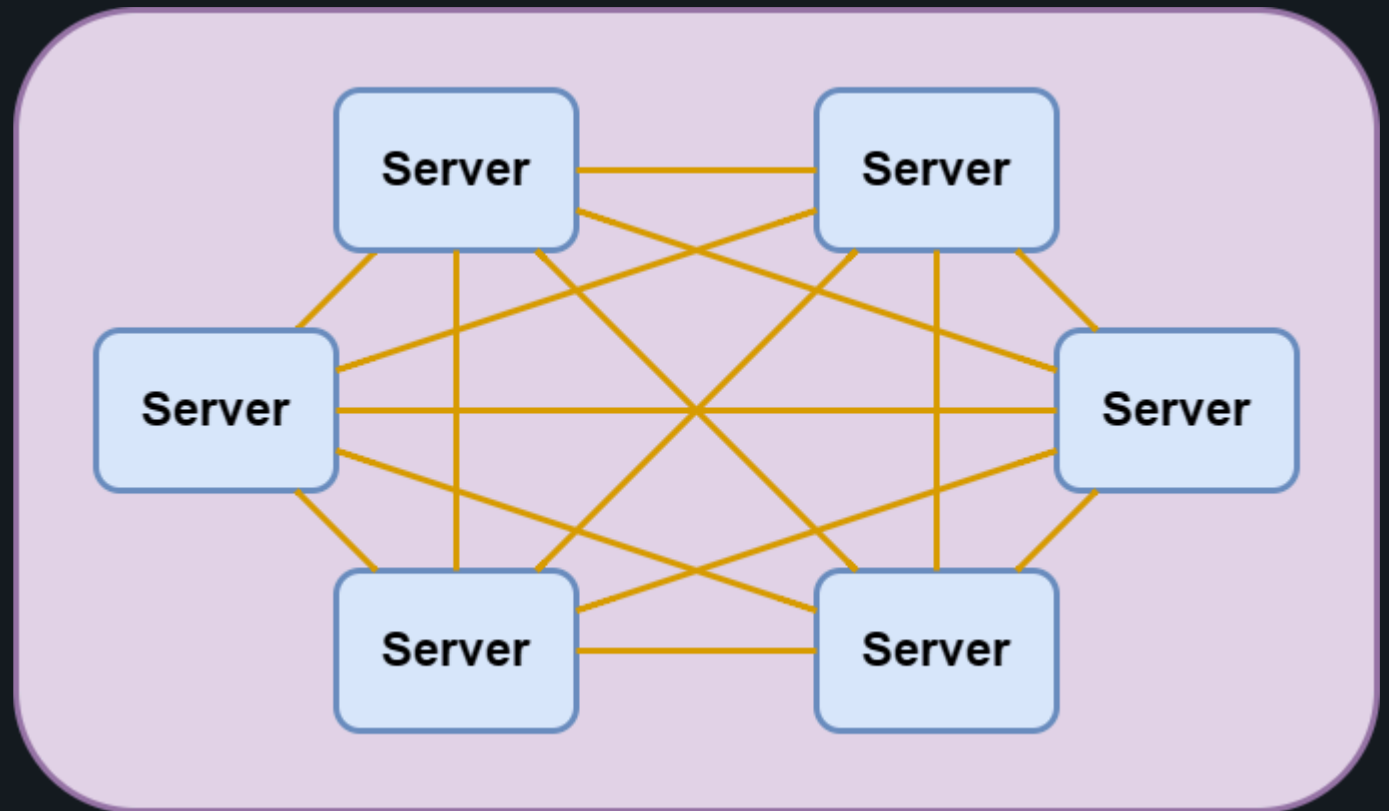
https://page.hyoo.ru/#!=5s0mg3_suuin9/#slide=1

Как выглядит наша система?

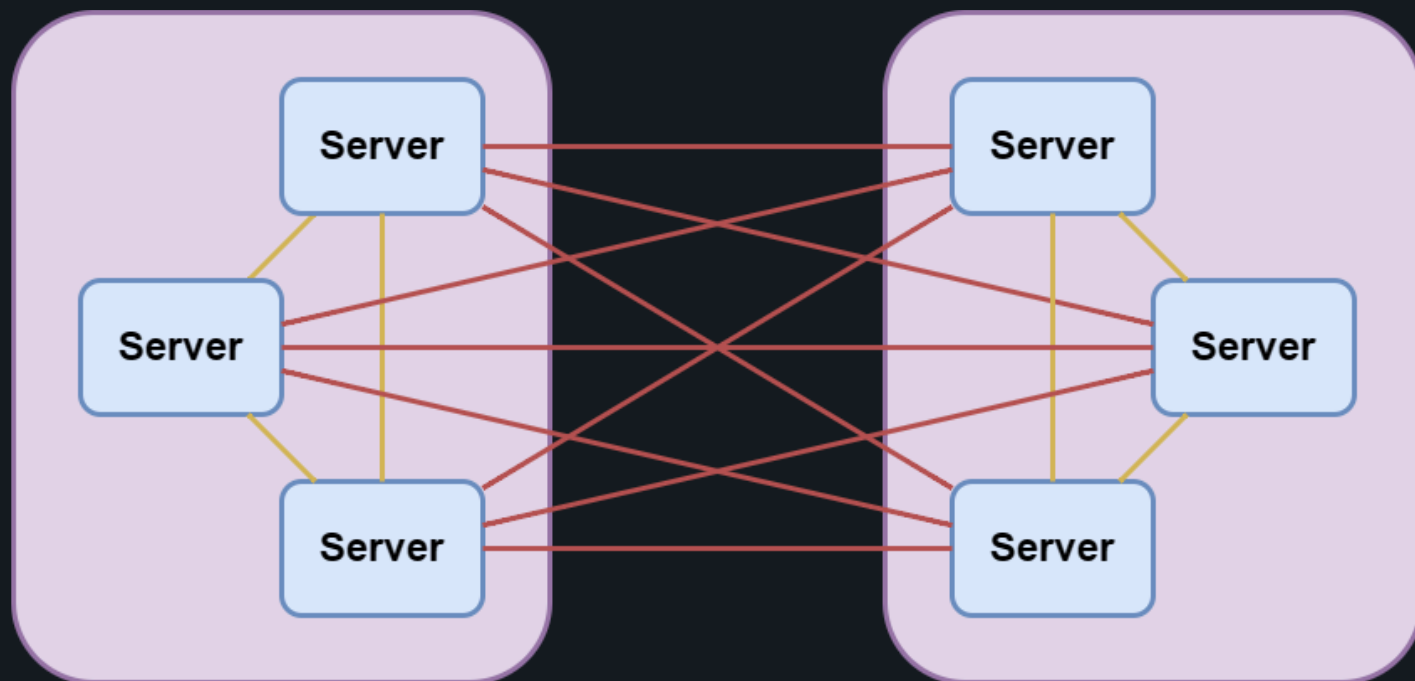
А если сервер упадёт?



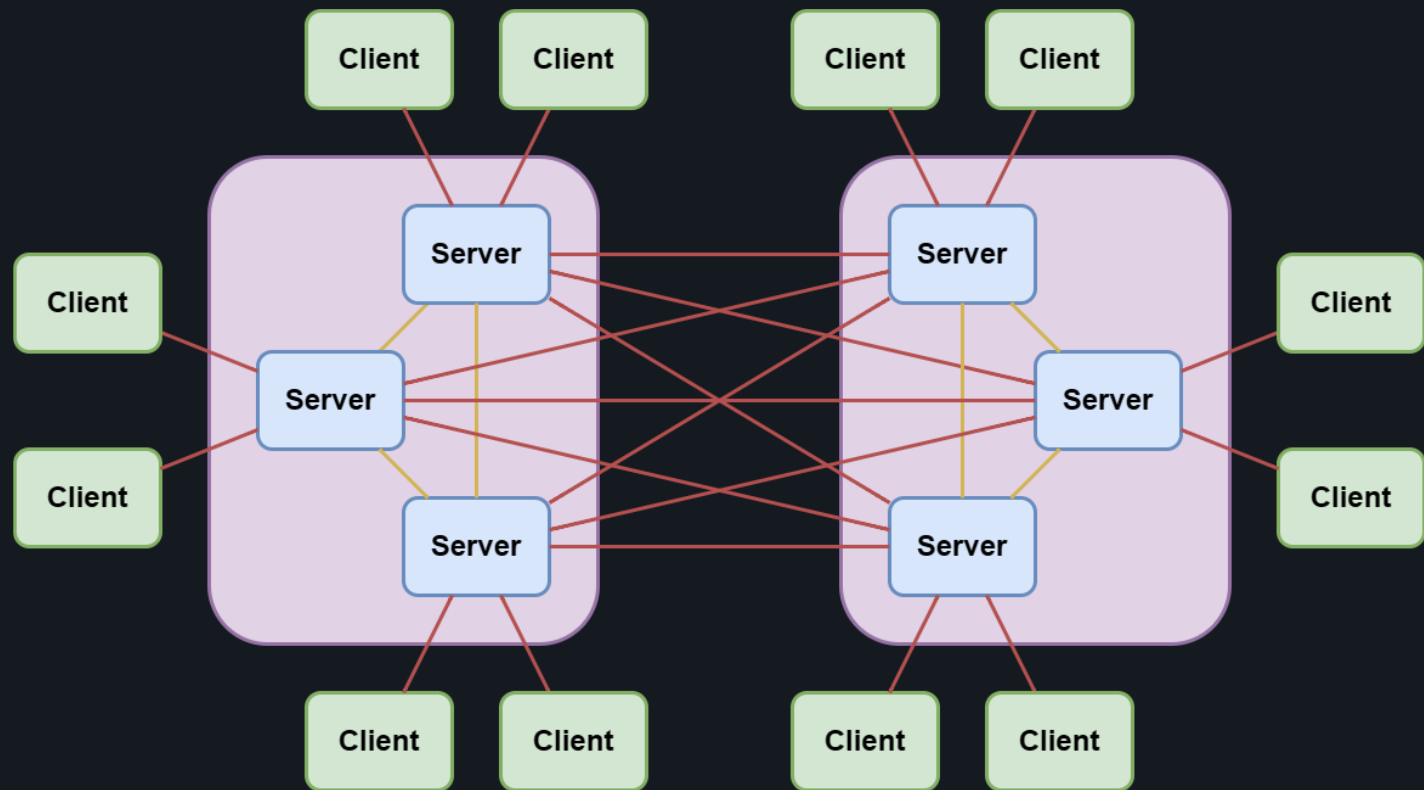
А если затопит дата-центр?



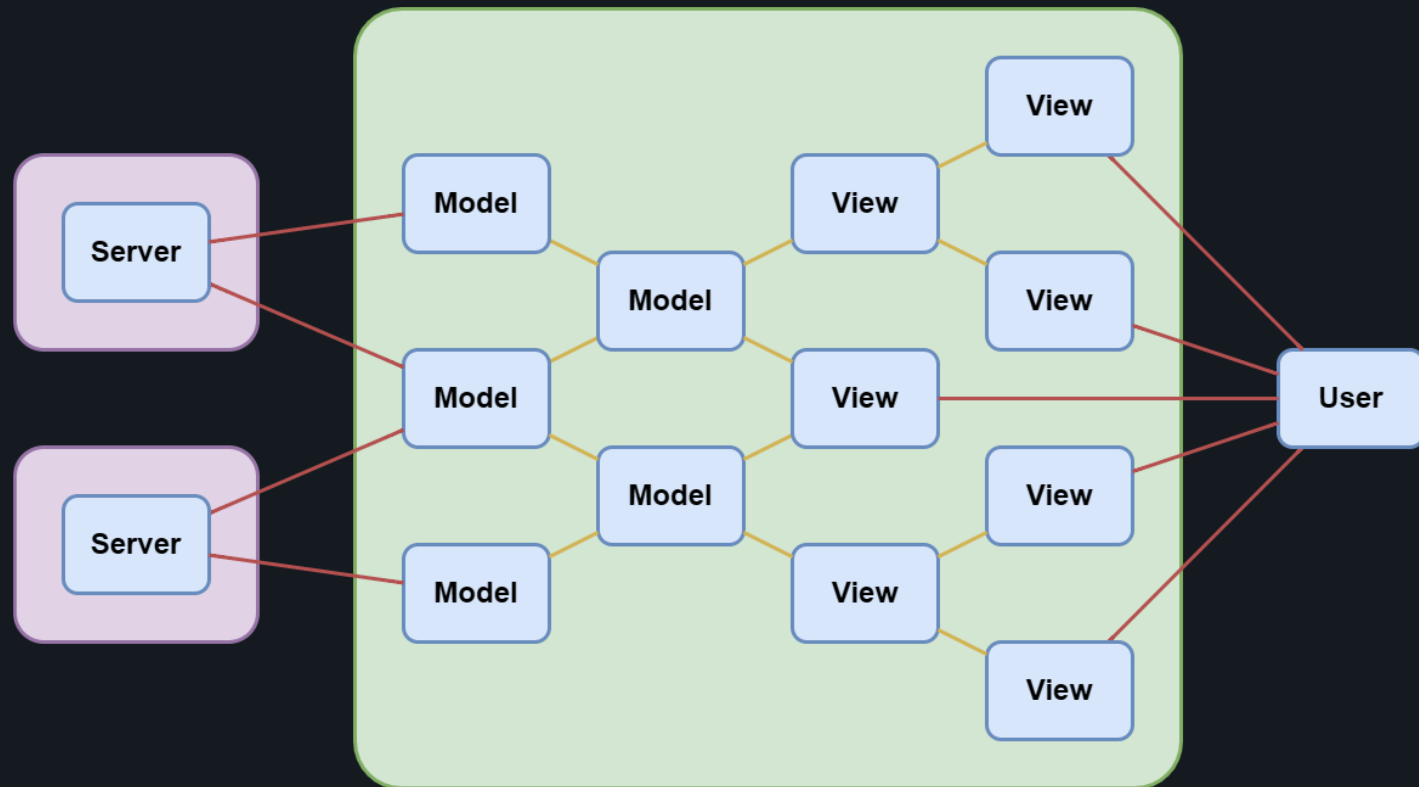
Но чего-то всё же не хватает..

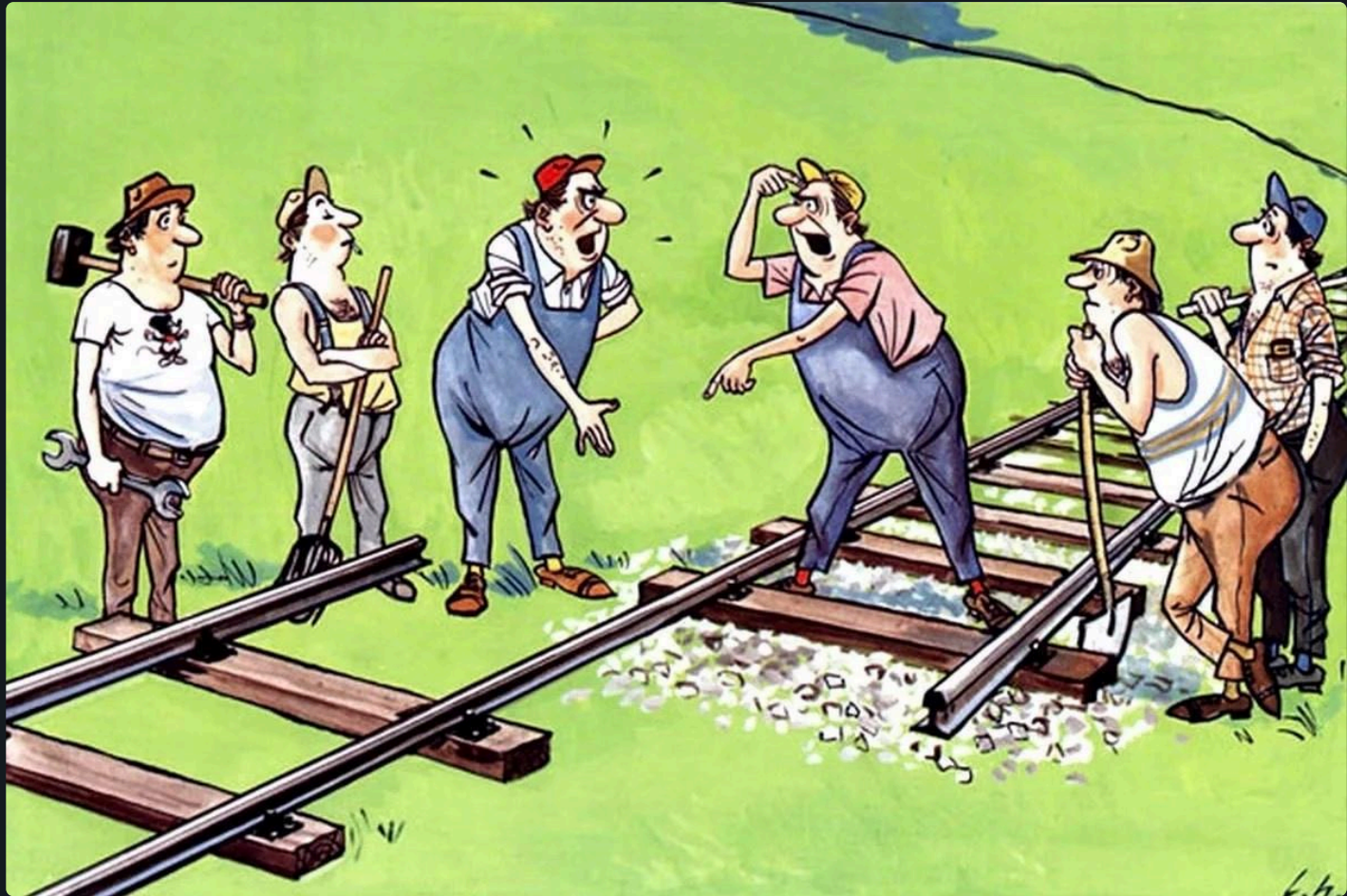


Но это только пол картины



Клиентское приложение - отдельная вселенная, но проблемы всё те же





Мифы и реалии «Мультимастера» в архитектуре СУБД PostgreSQL: 🦠, 🦠, 🦠.

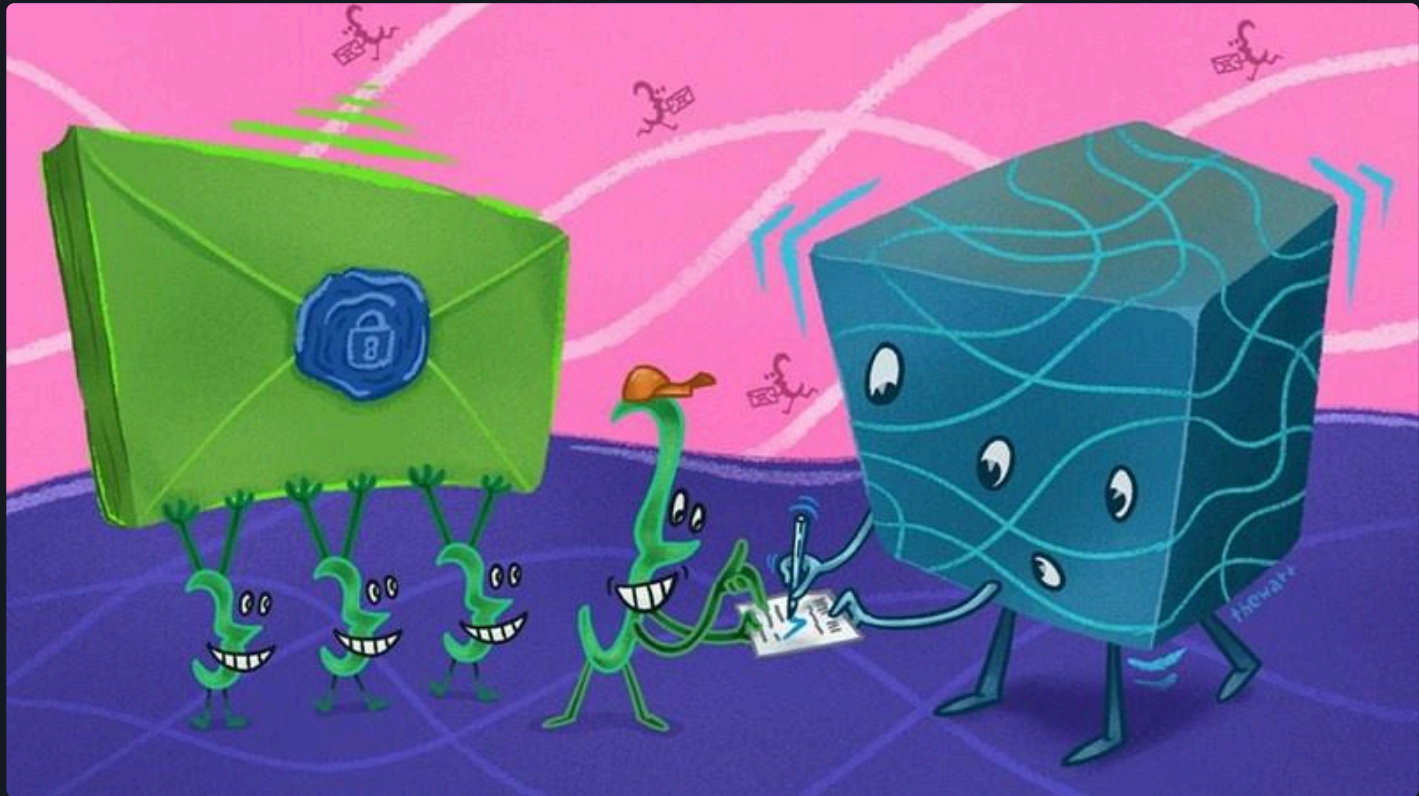


🌟 Событийная: событие существует в моменте

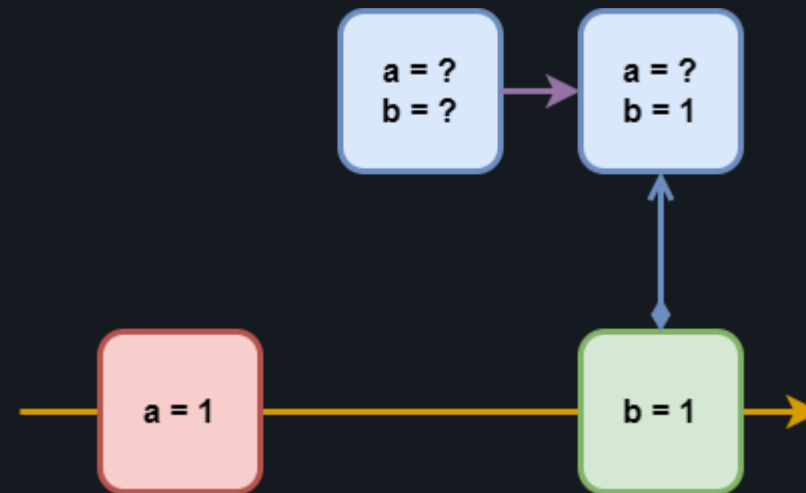
🚀 Реактивная: инварианты верны всегда

Гонка между подписками и
возникновениями

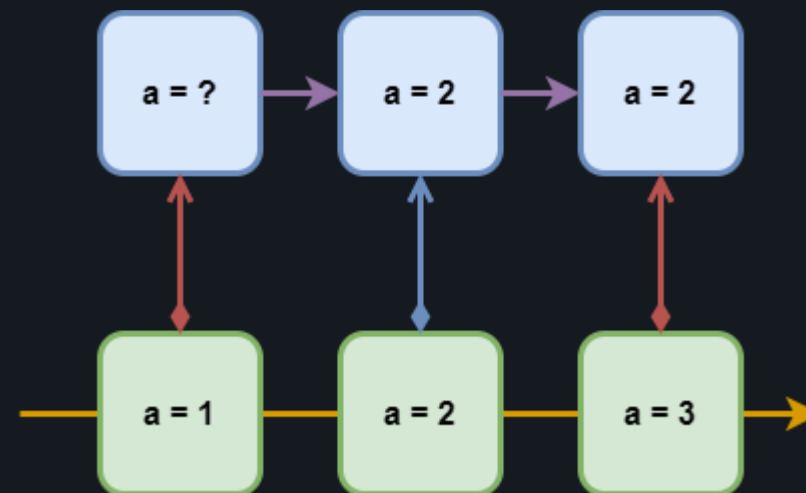
- ☛ Семантики доставки событий в распределенных системах / Avito
- ☛ Что может пойти не так у микросервисных приложений? / Alfa
- ☛ 4 вида распространённых ошибок в Event-Driven системах / OTUS



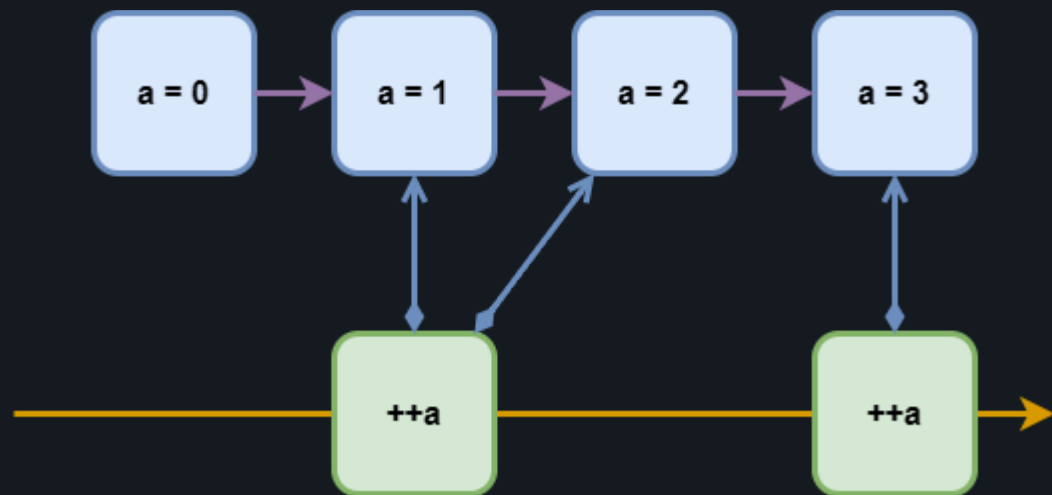
Не успел подписаться - событие не получил.



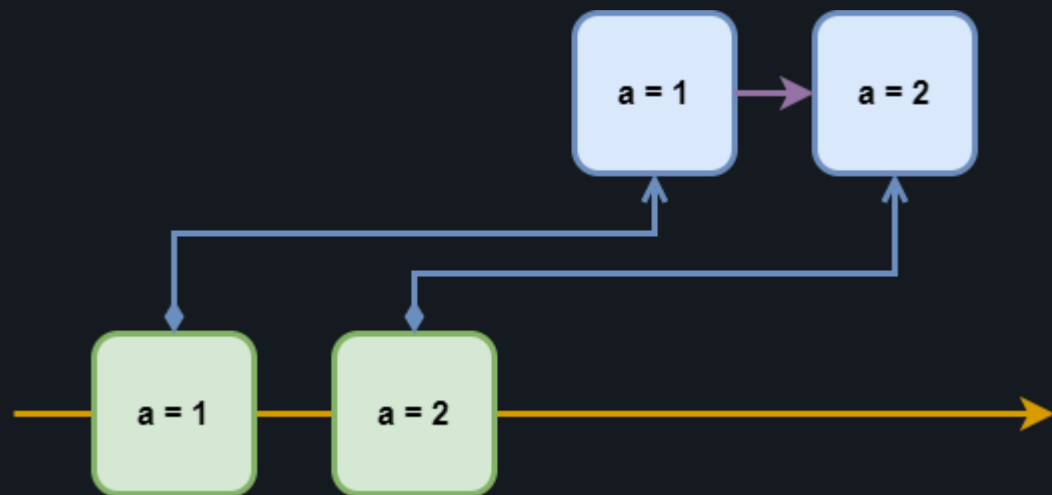
Не смог обработать - событие улетело.



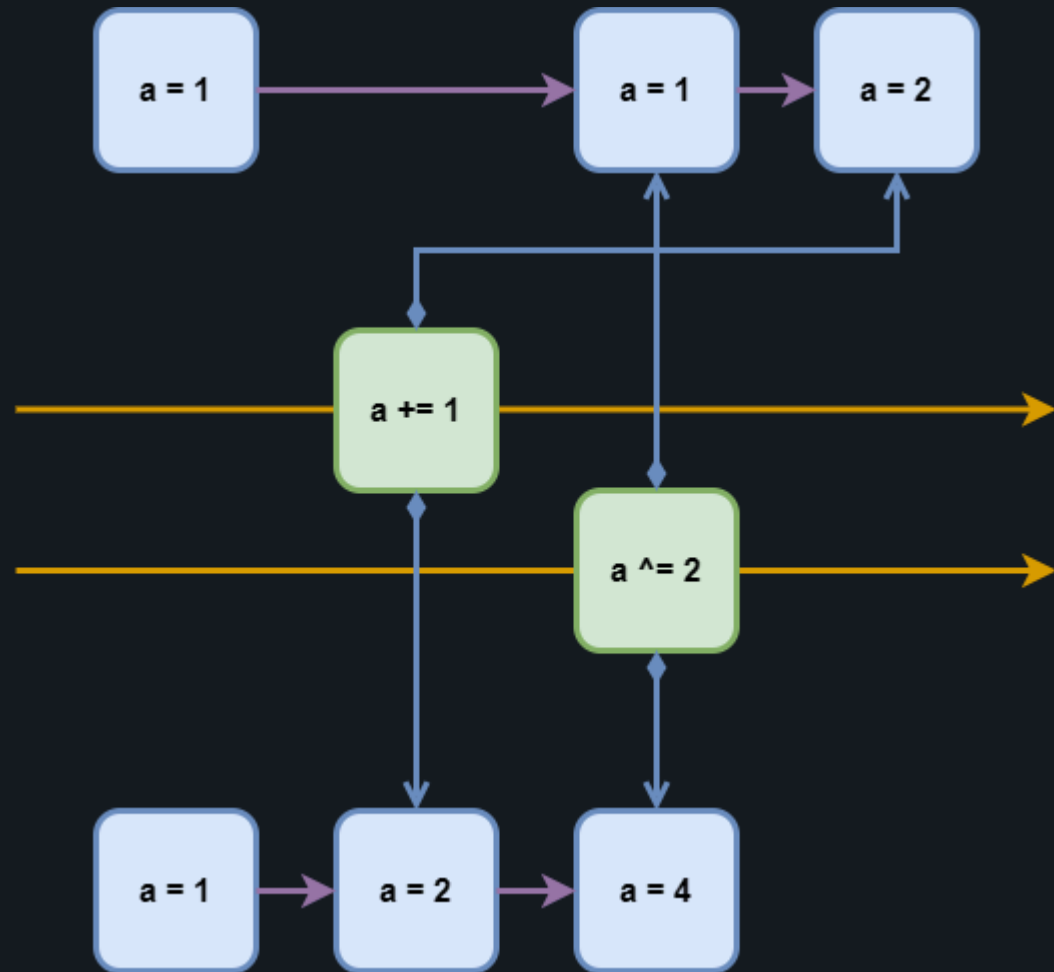
Не получилось подтвердить доставку -
получай дубликат.



Новое событие может делать обработку старого бессмысленной, но всё равно ждёт своей очереди.



Каждое событие само по себе. Могут прилетать в любом порядке.



Центральный брокер - единая точка отказа.
Репликация требует протоколов
консенсуса. А они не совместимы с
высокой доступностью.

W Paxos


🏠 Raft

W CTCA

Про Рахос из Вики: "Результат отказоустойчивого консенсуса не определён, однако, гарантируется безопасность, а условия, при которых консенсус невозможен, очень редки."

Используем события для голосования за правильный порядок событий.

 Raft (не)всемогущий / VK, Tarantool

 Повышаем живучесть Raft в реальных условиях / VK Tarantool

 The Saddest Moment / James Mickens

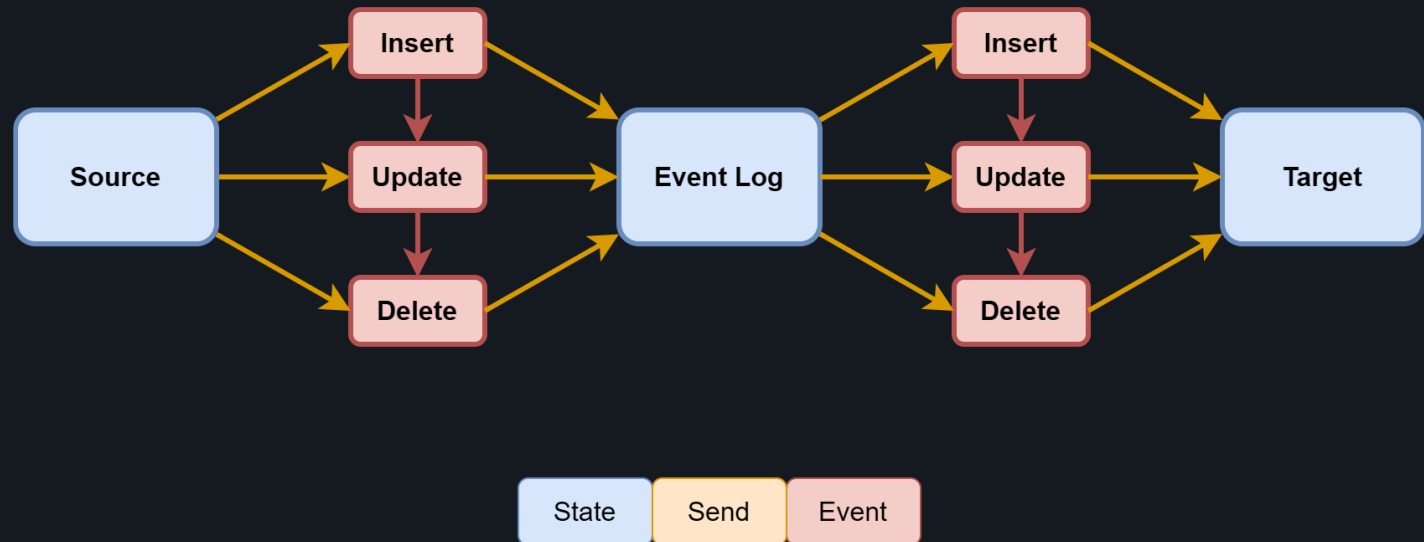
Различные брокеры сообщений:

- 🦋 Apache Kafka
- 🐙 Apache Pulsar
- 📡 MQTT
- 🐇 RabbitMQ

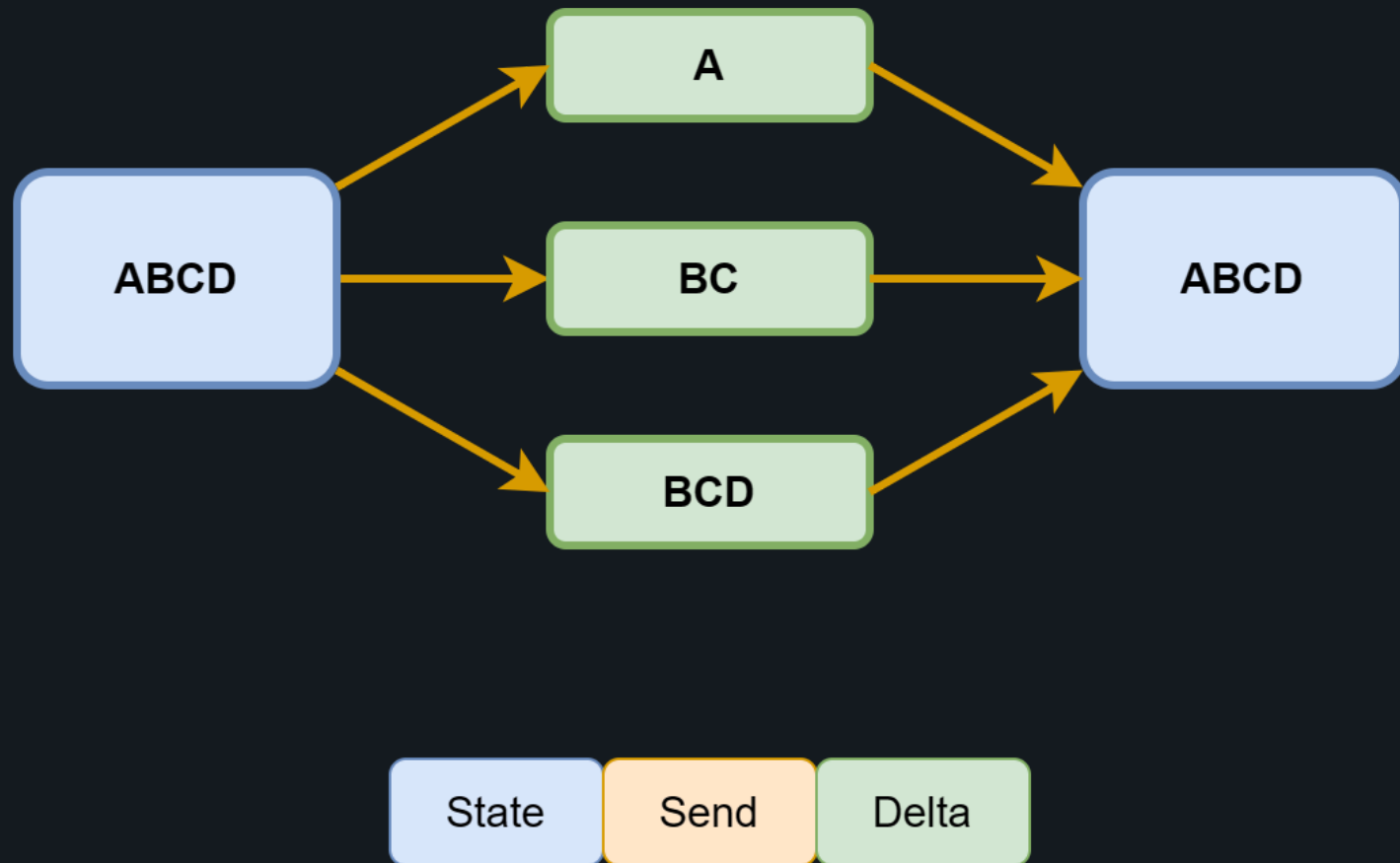
Преобразуем изменения состояния в события.

 Change Data Capture

Что-то тут не так, давай по новой!



Избавляемся от событий - решаем проблемы в корне.



- Работа с самым актуальным значением
- Подписка в любой момент
- Спокойно падаем и перезапускаемся
- Спокойно реконнектимся и доставляем недостащу
- Обновления сразу пачками

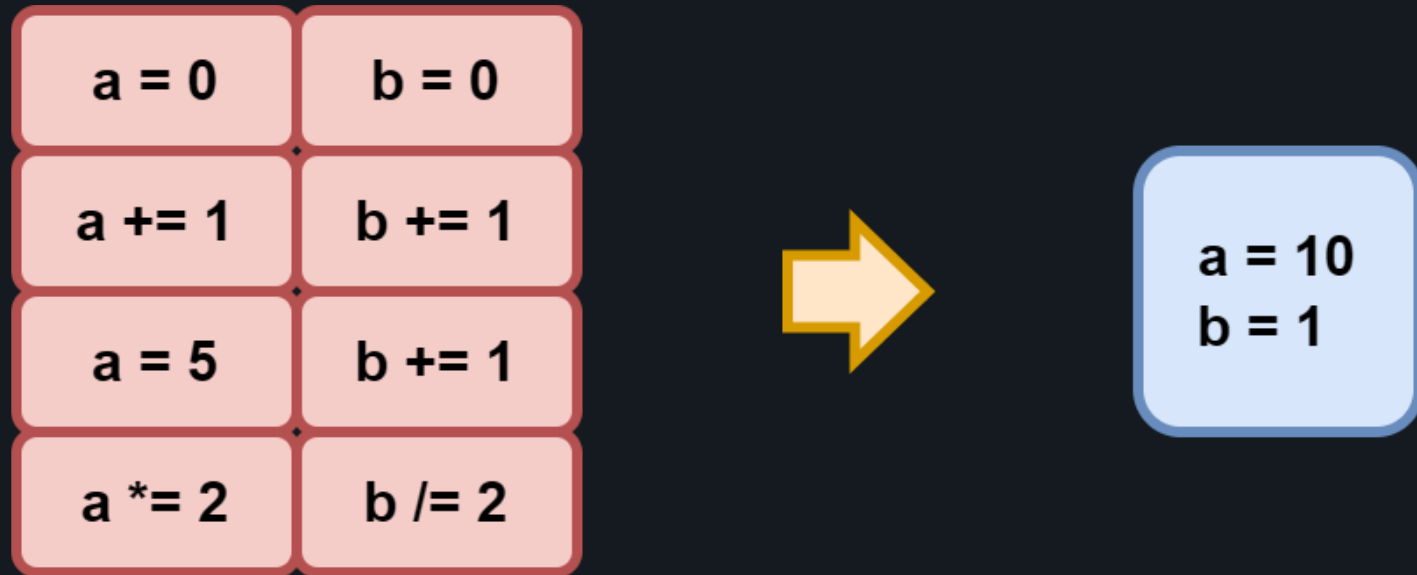
- Слепки состояния (LWW)
- Логи событий (ES/OT/CmRDT)
- Набор юнитов (CvRDT)

*полное состояние на определённый
момент*

a = 10
b = 1

State

все события изменения с начала времён, из которых восстанавливается состояние

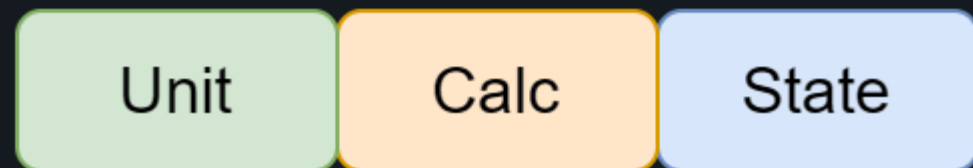


Event

Calc

State

*изменяемый набор минимальных единиц
смысла, из которых восстанавливается
состояние*



для примера, описание упрощённого Хабра
- статьи с мета инфой

Слепки	Полный текст статьи
События	Лог всех правок статьи
Юниты	Частично упорядоченный набор слов

определяет накладные расходы на хранение и передачу

Слепки	✓ размер_данных
События	✗ размер_данных * E + число_изменений * H
Юниты	○ размер_данных * M

*Слепки дорого обновлять на каждый чих, а
из событий дорого восстанавливать
финальное состояние*


Слепки	✗ Медленная работа
События	✗ Медленный старт
Юниты	✓ Всё шустро

Оптимистичное изменение слепков вызывает глич, а пессимистичное - задержки, слепки пересылать дорого на каждый чих

	Загрузка	Изменение	Пакеты
Слепки	✓ Быстро	✗ Глич ✗ Лаги	✗ Большие
События	✗ Долго	✓ Мгновенное	✓ Малые
Юниты	○ Умеренно	✓ Мгновенное	○ Умеренные

Наблюдение за промежуточным состоянием может вызывать каскад последствий которые крайне сложно, а то и не возможно корректно откатить.

 Read Uncommitted

 Realistic UI: реалистичный взгляд на Optimistic UI

 Пиррова победа Domain-Driven Design

для борьбы с конфликтами, все изменения идут по максимальной дуге, собирая по пути множество задержек

Конфликты при параллельных изменениях - просто беда. Традиционный подход с хранением состояния в виде слепков принципиально не решает этих проблем на разных уровнях:

W Unidirectional Data Flow

- синхронизация инпута и стейта

W Single Source of Truth

- синхронизация между клиентом и сервером

W Master-Slave Replication

- синхронизация между серверами

Центральное звено - узкое место

- ✗ Сервер может тормозить
- ✗ Сервер может упасть
- ✗ Сервер могут заблокировать
- ✗ Сервер могут выключить
- ✗ Сервер не доступен в оффлайне
- ✗ Облако не доступно в изолированном периметре

Повышенные требования к безопасности сервера, которых всё равно не достаточно

- ✗ Может прочитать и слить любые данные
- ✗ Может опубликовать что-либо от твоего имени
- ✗ Может удалить твои данные
- ✗ Может тебя капитально забанить

Что изначально открыто - слито быть не может. А закрытие обеспечивается криптографией.

- ✗ Производитель железа
- ✗ Хостер сервера
- ✗ Хитрый хакер
- ✗ Случайный троян (привет, СДЭК)
- ✗ Коварный админ
- ✗ Любой сотрудник (привет, Яндекс Еда)

*Что если доступ к серверу не будет
давать никаких привилегий*

- ✓ Админ сервера не имеет доступа к приватным данным
- ✓ Админ сервера не может подделать данные
- ✓ Можно использовать ненадёжные каналы связи
- ✓ Возможно настоящее P2P

Админы делятся на следующие типы:

- 🤪 Кто ещё не делает бэкапы
- 🧐 Кто уже делает бэкапы
- 😎 Кто проверяет, что бэкапы рабочие
- 😬 Кто уже не делает бэкапы и грохает базу по приколу

изменения мгновенно вносятся локально и даже в оффлайне, а потом отложено синхронизируются, не теряя локальные изменения

🌐 Local-First

🌐 Multi-Master Replication

🌐 Zero-Trust Security Model

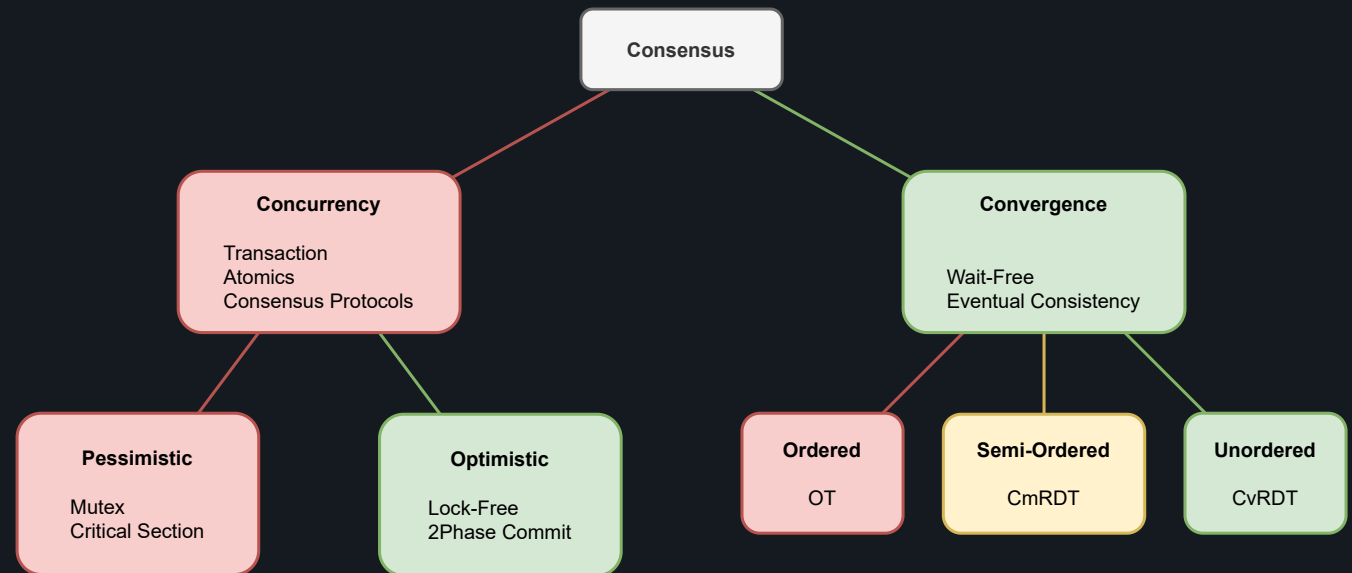
*Нельзя просто так взять любой CRDT,
нужно выполнение кучи требований*

- Без конфликтов изменений: ✗ REST, ✗ GQL, ✗ OData, ✗ Flux...
- С цифровыми подписями и без изменения юнитов: ✗ OT, ✗ YATA...
- Без дешифрации данных для слияния: ✗ GIT, ✗ SVN, ✗ HG...
- Консенсус через конвергенцию: ✗ Blockchain, ✗ PAXOS, ✗ Raft...
- С криптографическим контролем прав

Архитектура крайне простая, но готовых реализаций почти нет - огромный простор для велосипедов

- Все изменения применяются сразу и не теряются
- Не нужны очереди сообщений для отказоустойчивости
- Не нужны сложные голосования для консенсуса
- Синхронизация дельтами в реальном времени

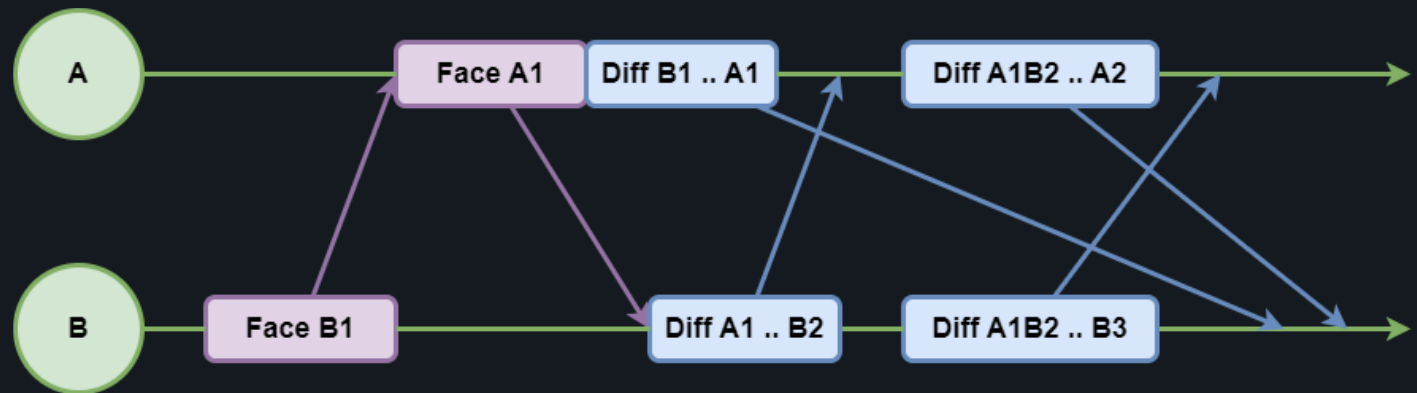
Консистентно о Консенсусе



- Деревья, документы, словари
- Таблицы (строковые, колоночные)
- Графы, объекты

- Граф из виртуальных узлов, но материальных рёбер.
- Юнит - кирпичик информации: Кто, Где, Когда, Что, Пруф.
- Ленд - единица авторизации и синхронизации.

Крайне простой, но надёжный. При соединении обмениваемся **часами**. Шлём **дельты** без подтверждения. При обнаружении утраты - полный **ресинк**.



CvRDT


Граф с группировкой в кластеры синхронизации

Динамическая типизация без потери слияния

Бесконфликтное слияние без дешифровки

Локальное хранение в браузере: IndexedDB


Хранение на сервере: NodeJS + FS/PG


Реактивность через  \$mol_wire


TS библиотека (50KB)

- Самовосстановление данных
- Полная работоспособность оффлайн
- Синхронизация в реальном времени


- Большой размер мета-данных (~10x)
- Ресурсоёмкая криптография
- Никаких серверных миграций

 crus.hyoo.ru - база данных

 mol.hyoo.ru - фреймворк

 [mam_mol](#) - про вебдев

 [mam_mol](#) - новости

 [h_y_o_o](#) - обсуждения

 [hyoo](#) - донаты

 [nin_jin](#) - личка