10. Причинность, часы и версии данных

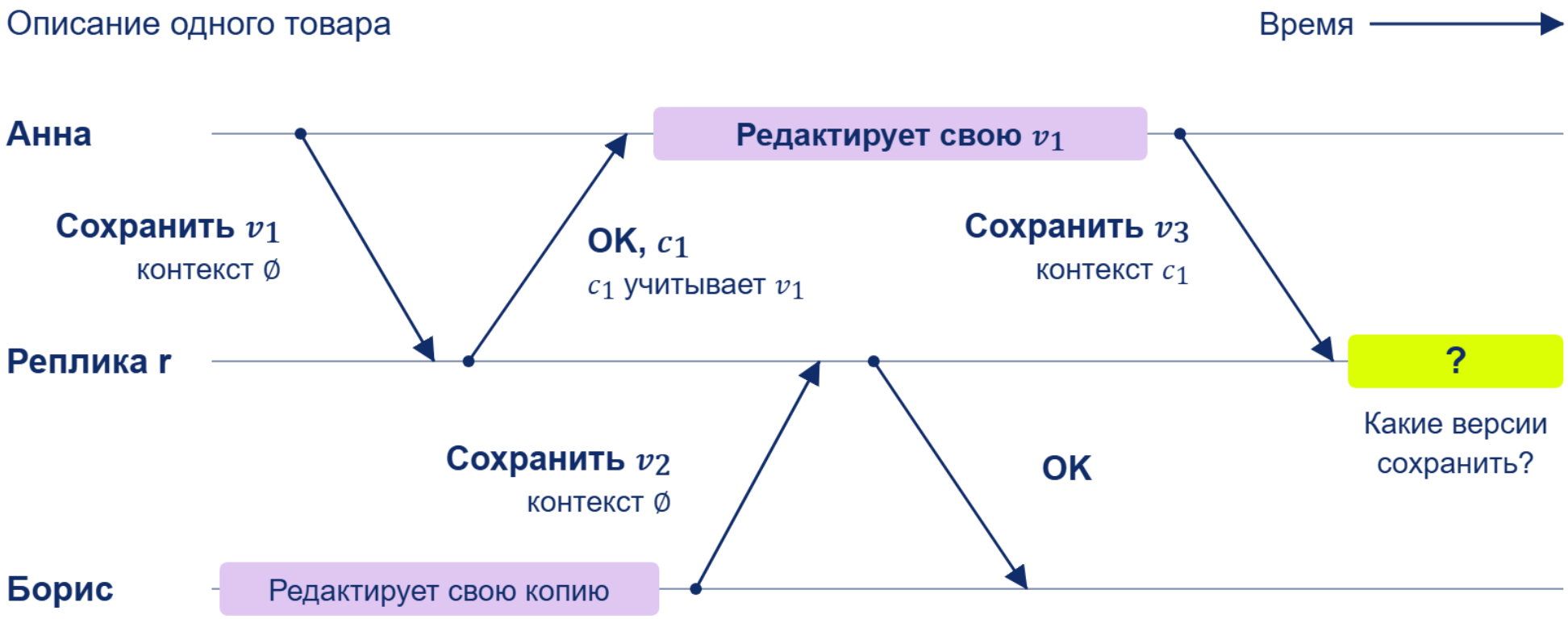
Сухорослов Олег Викторович

16.11.2026

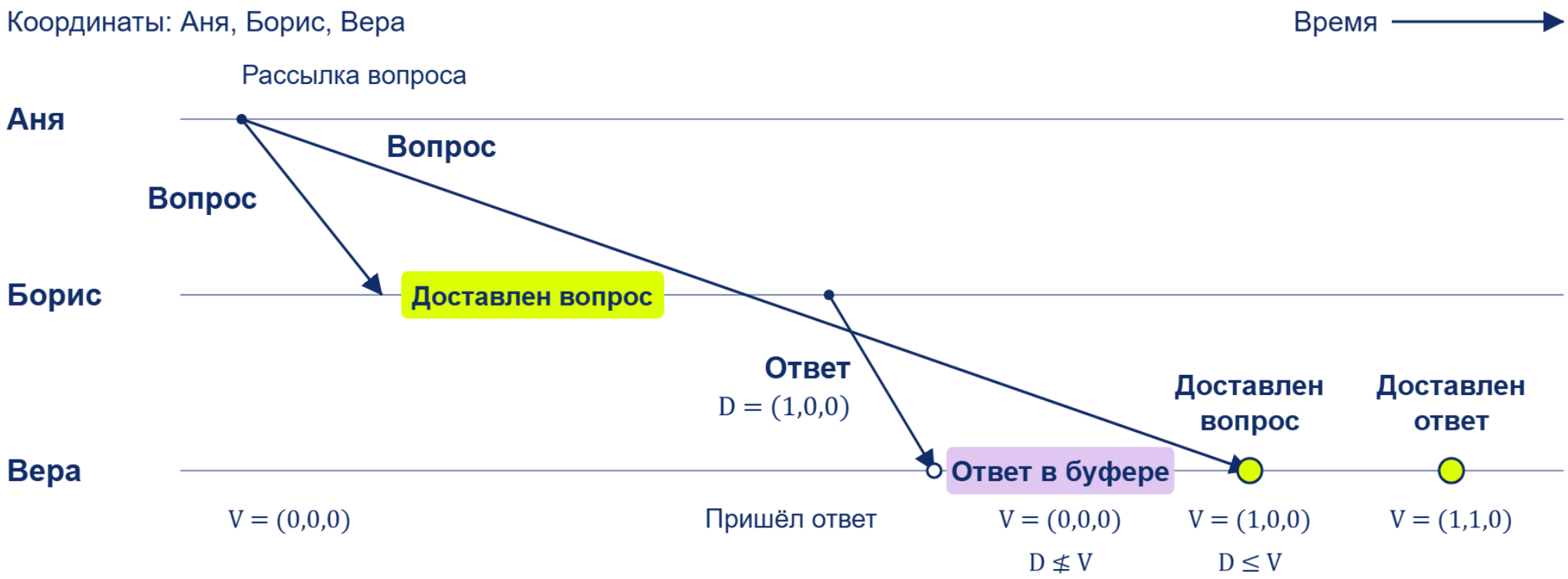
Два клиента редактируют одну запись

Хранилище принимает независимые правки и сохраняет конфликтующие версии.

Новая запись заменяет версии, учтенные её *контекстом*.



Причинная рассылка из лекции 4



D — сколько первых сообщений каждого отправителя должно быть доставлено перед текущим.

V — сколько сообщений каждого отправителя уже доставлено приложению.

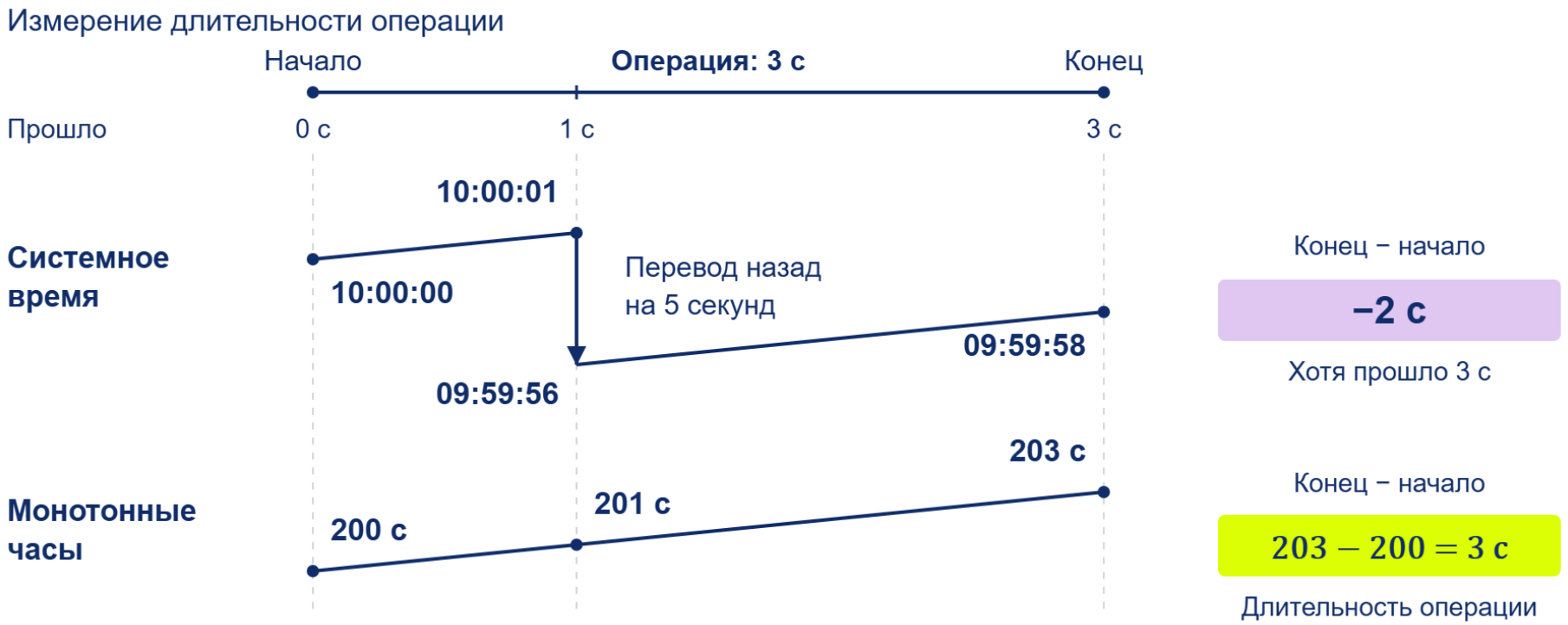
Сообщение можно доставить приложению, когда D ≤ V покомпонентно.

План лекции

- Физические часы и синхронизация времени.
- Причинность и модели согласованности.
- Логические часы: часы Лэмпорта и векторные часы.
- Причинная доставка сообщений и видимость зависимых записей.
- Векторы версий и ограничения простых меток версий.
- Слияние конкурентных изменений и CRDT.

Локальные часы

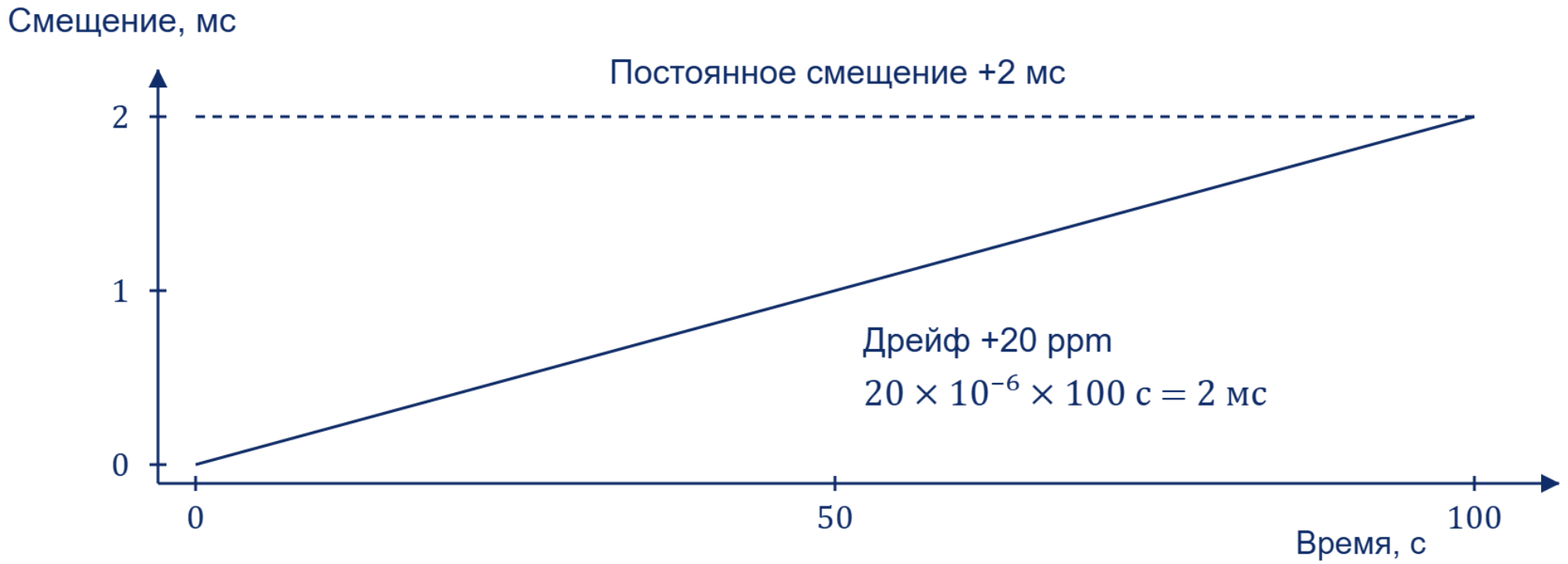
Вид часов	Использование
Часы системного времени (wall clock)	Дата и время события
Монотонные часы (monotonic clock)	Длительности операций и локальные тайм-ауты



Смещение и дрейф

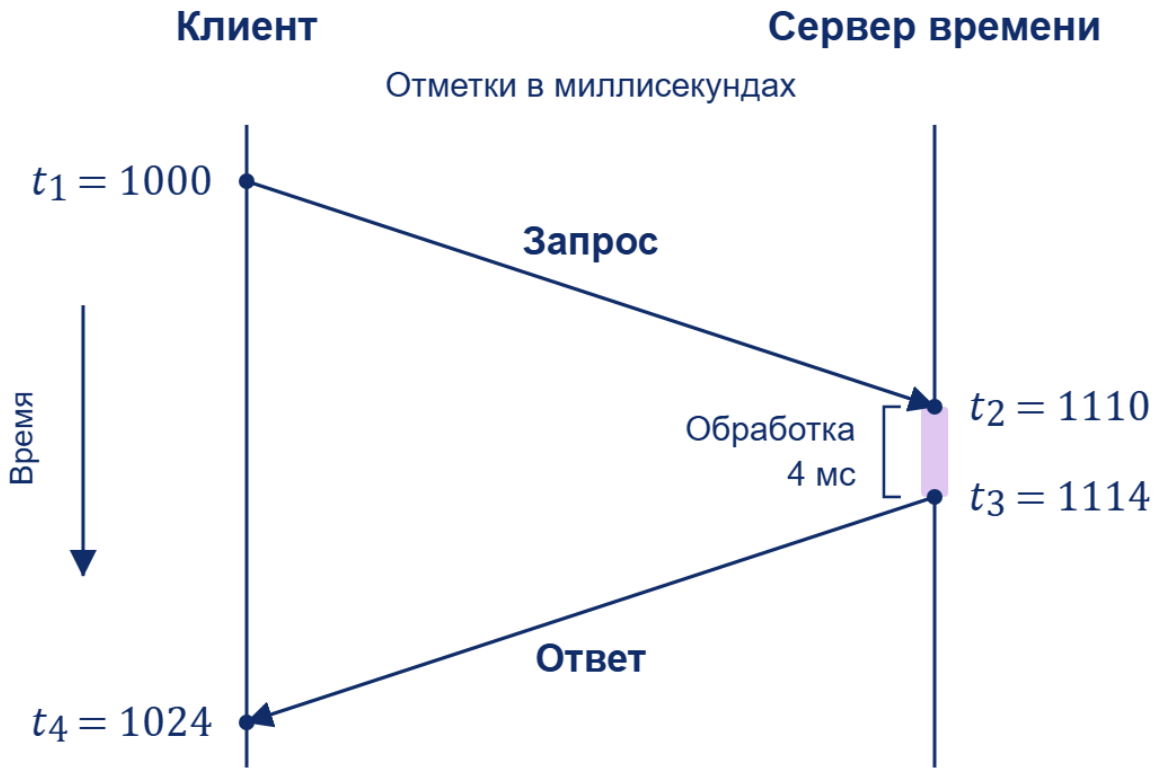
$C(t)$ — показание локальных часов в физический момент t.

Смещение: $C(t) - t$. **Дрейф:** отличие скорости хода часов от эталонной.



Измерение смещения часов в NTP

NTP (Network Time Protocol) — протокол синхронизации системных часов по сети.



Сетевая задержка

туда и обратно

$$\delta = (t_4 - t_1) - (t_3 - t_2)$$

$$\delta = 24 - 4 = 20 \text{ мс}$$

Смещение часов

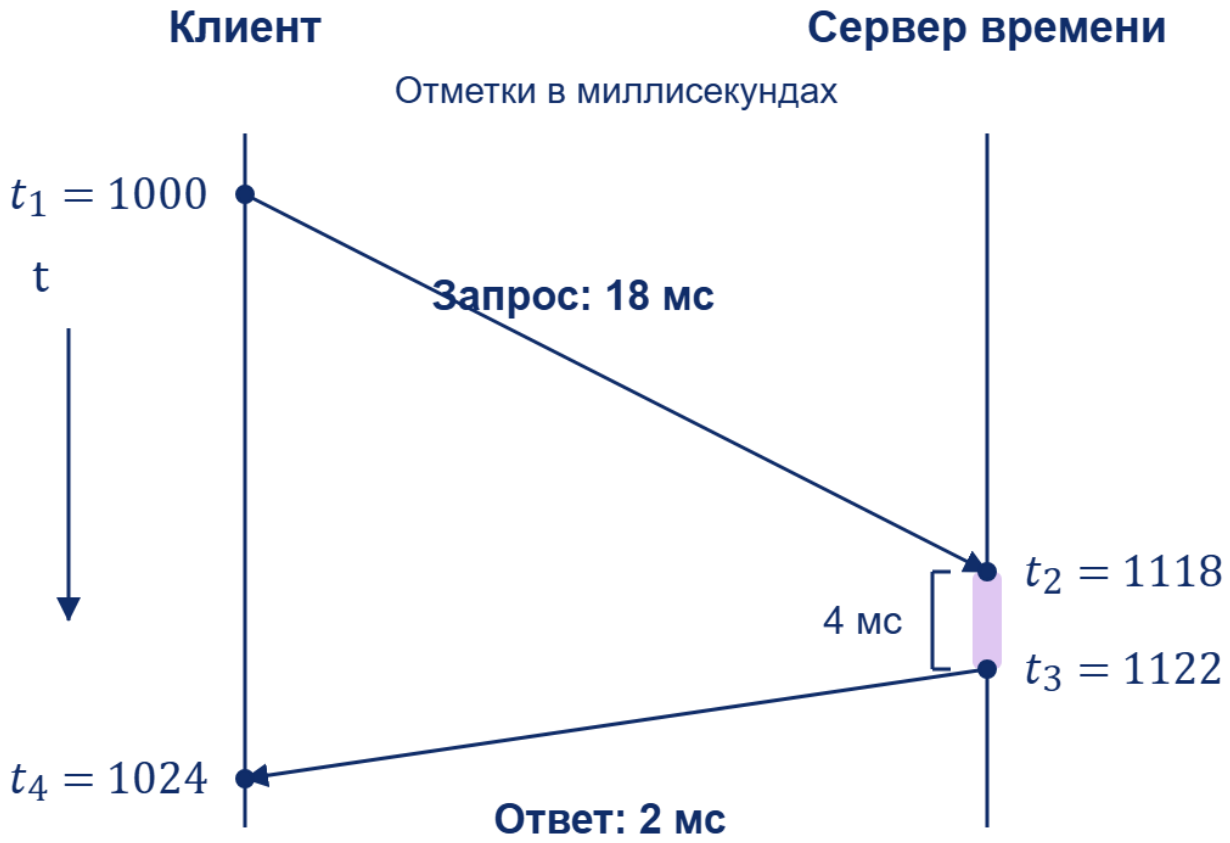
сервера относительно клиента

$$\widehat{\theta} = \frac{(t_2 - t_1) + (t_3 - t_4)}{2}$$

$$\frac{110 + 90}{2} = 100 \text{ мс}$$

Оценка смещения предполагает равные задержки запроса и ответа.

Асимметрия сетевой задержки



Истинное смещение

$\theta = 100$ мс

Сетевая задержка

$\delta = 18 + 2 = 20$ мс

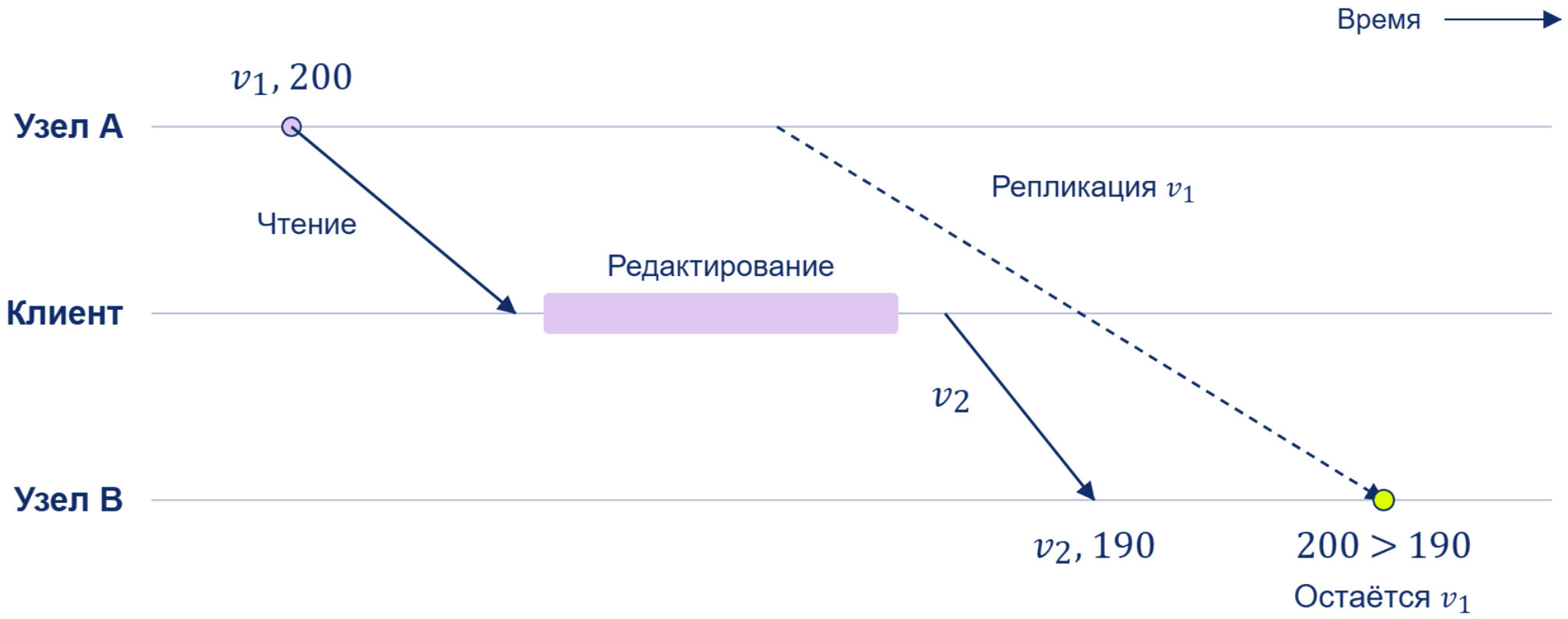
Оценка смещения

$$\hat{\theta} = \frac{118 + 98}{2}$$

108 мс: ошибка +8 мс

Ошибка оценки равна половине разности задержек запроса и ответа.

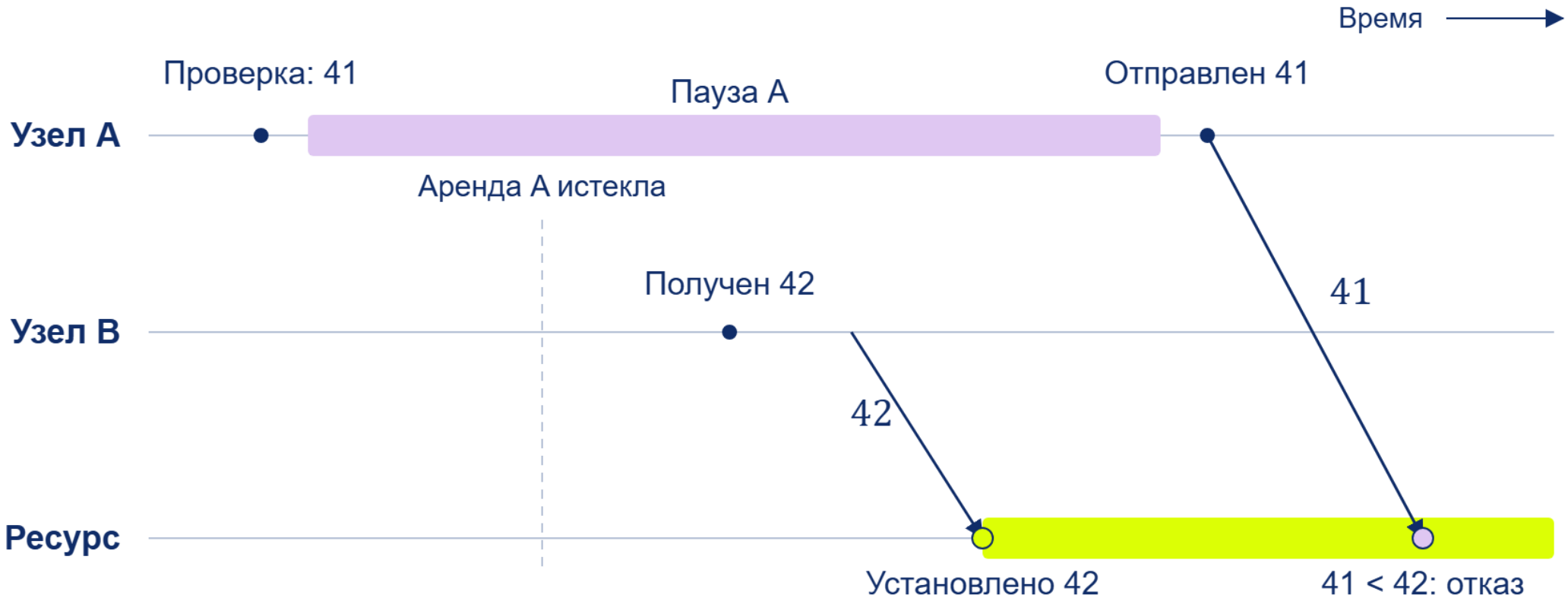
Last Write Wins и отстающие часы



Метка записи берётся напрямую из физических часов принимающего узла.

LWW (Last Write Wins) выбирает v_1, хотя v_2 подготовлена на её основе.

Пауза после проверки lease

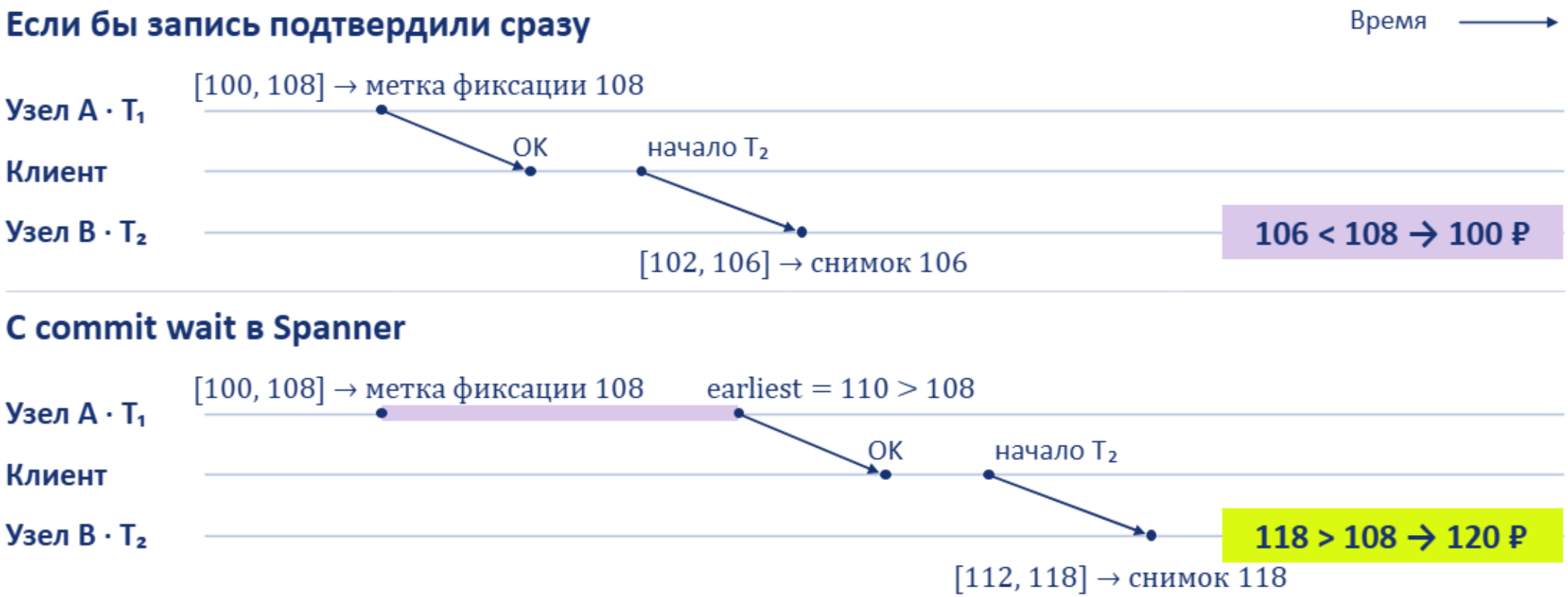


Fencing отклоняет старые токены после установки нового поколения на ресурсе.

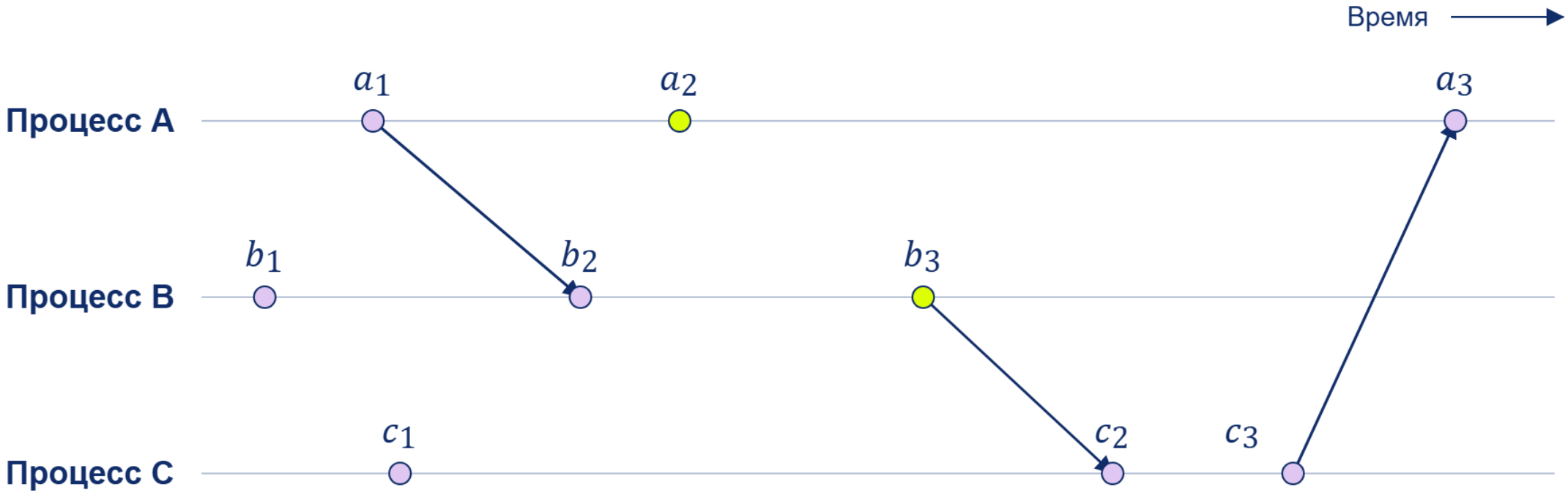
TrueTime и видимость записи в Spanner

TrueTime возвращает интервал $[earliest, latest]$, содержащий текущее время.

T_1 меняет цену со 100 до 120 ₽. Клиент запускает чтение T_2 после подтверждения T_1.



События и связи между ними

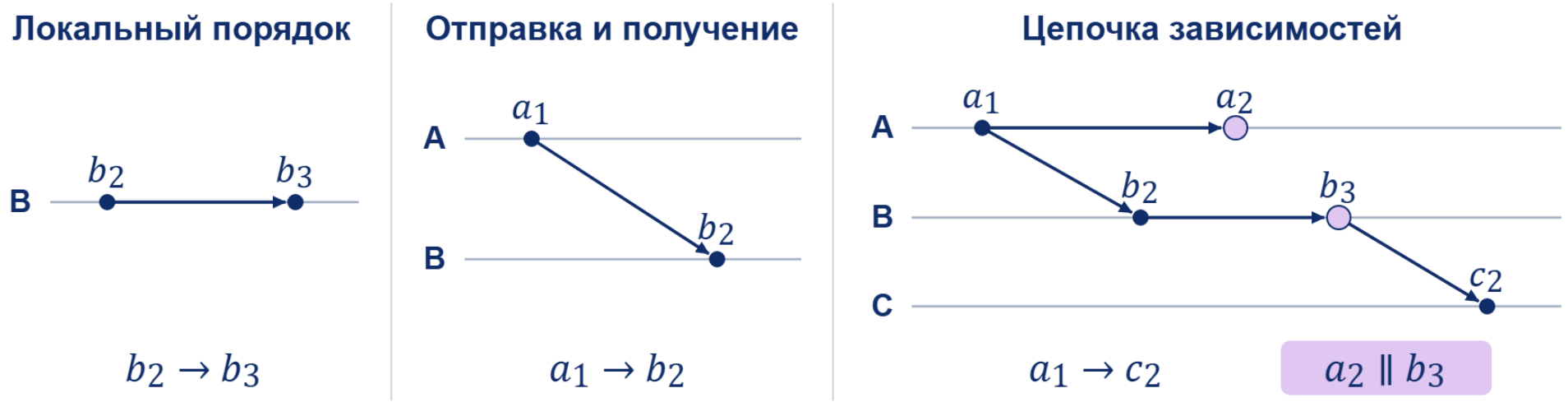


События: локальное действие, отправка или получение сообщения.

Горизонтальная линия задаёт локальный порядок событий внутри процесса.

Стрелка связывает отправку сообщения с его получением.

Отношение happened-before

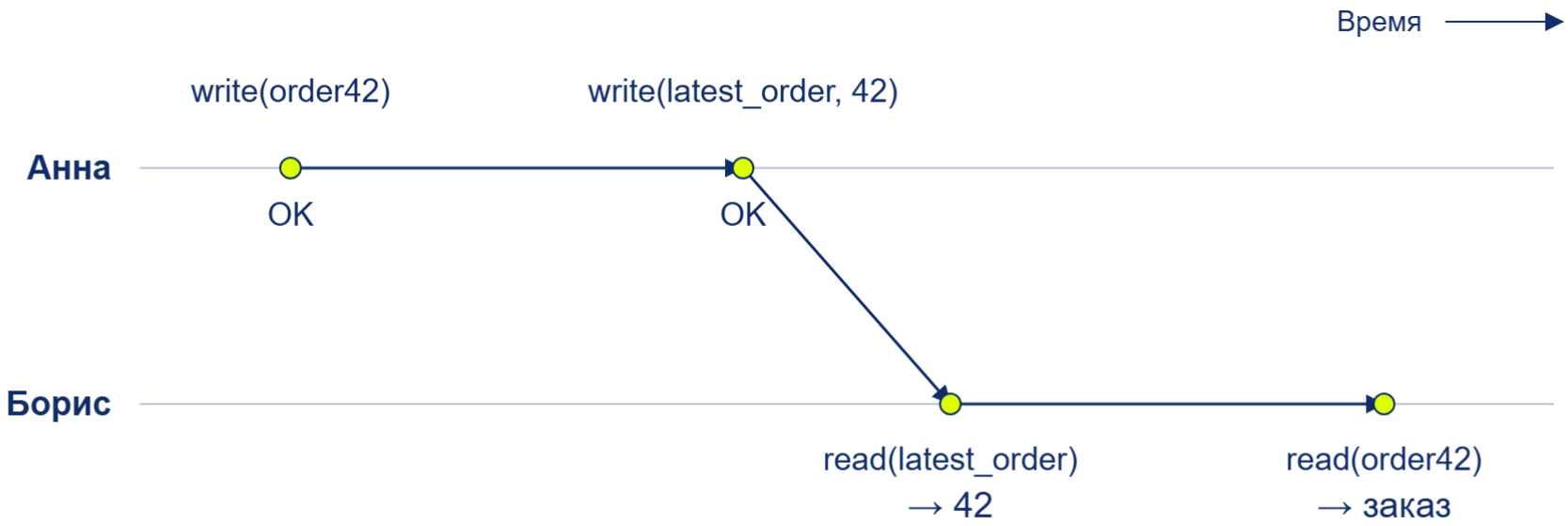


$a \to b$, если выполняется одно из трёх условий:

- События принадлежат одному процессу и a раньше b.
- a — отправка сообщения, b — его получение.
- Существует c, для которого $a \to c$ и $c \to b$.

Если для разных событий нет пути ни в одну сторону, они конкурентны: $a \parallel b$.

Причинная согласованность



Зависимости операций задаются:

- порядком операций одного клиента;
- связью «запись → чтение её результата»;
- транзитивностью.

Для каждого клиента существует допустимая последовательность всех записей и его операций, сохраняющая эти зависимости.

Порядок конкурентных записей у разных клиентов может различаться.

Требования к порядку операций

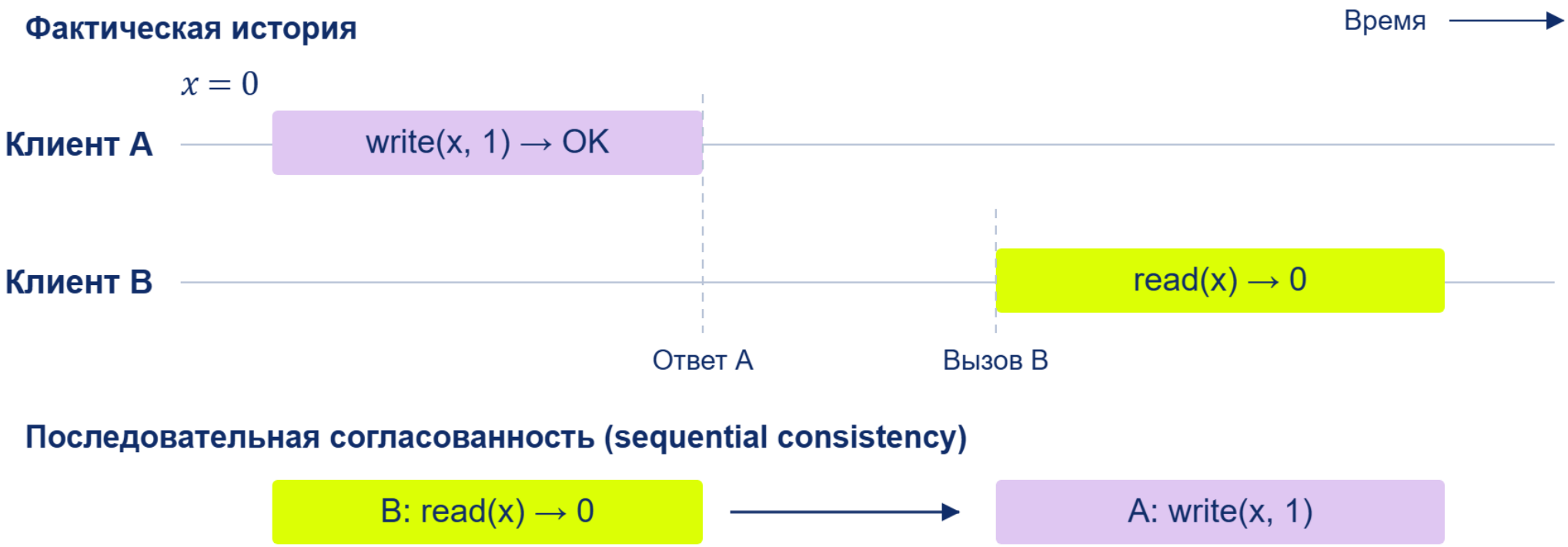
Модель согласованности	**Требование**
Линеаризуемость	Общая допустимая последовательность с сохранением *порядка реального времени*
Последовательная согласованность (sequential consistency)	Общая допустимая последовательность с сохранением *порядка операций каждого клиента*
Причинная согласованность (causal consistency)	Сохранение *причинных зависимостей*; порядок конкурентных записей может различаться

Каждый клиент выполняет чтения и записи регистров последовательно.

В допустимом порядке чтение возвращает значение последней записи своего ключа.

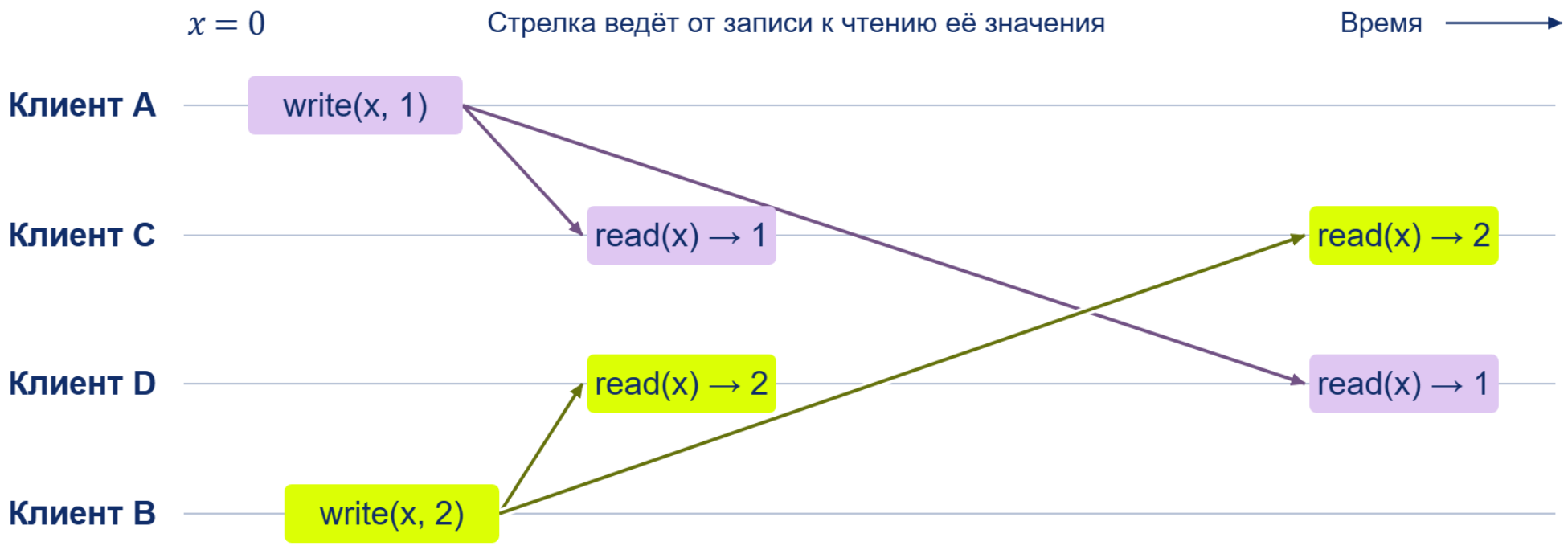
Если записи ещё не было, чтение возвращает начальное значение.

Порядок клиента и реальное время



История не линеаризуема, но допустима в рамках модели sequential consistency.

Разный порядок конкурентных записей



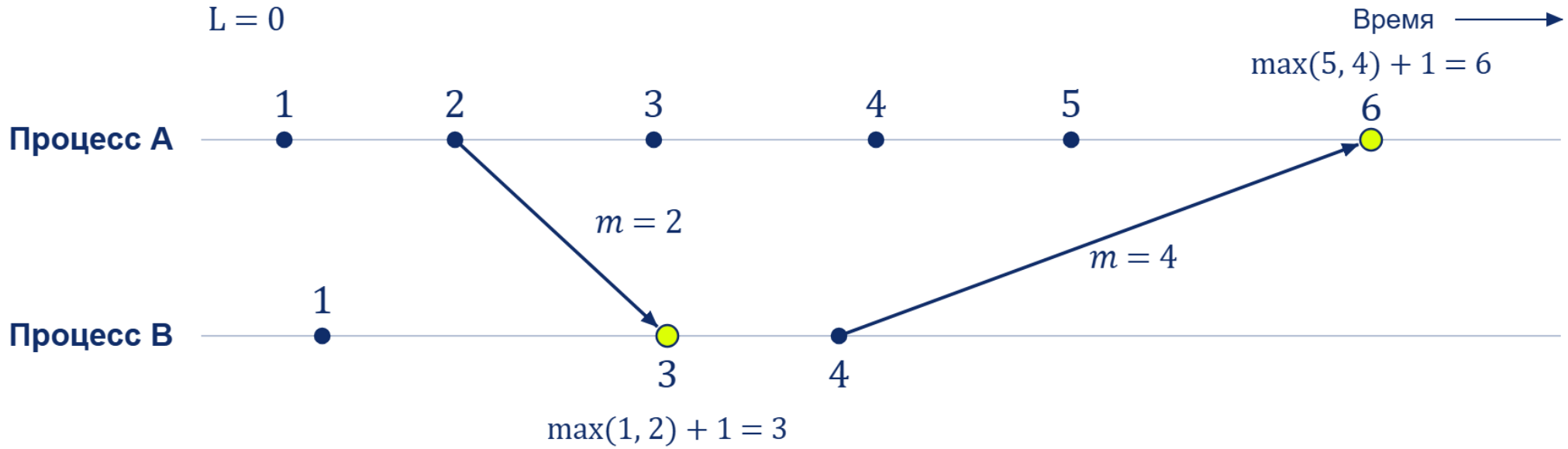
История нарушает sequential consistency (нет единого порядка операций),
но допустима в рамках причинной согласованности (операции записи конкурентны).

Логические часы Лэмпорта

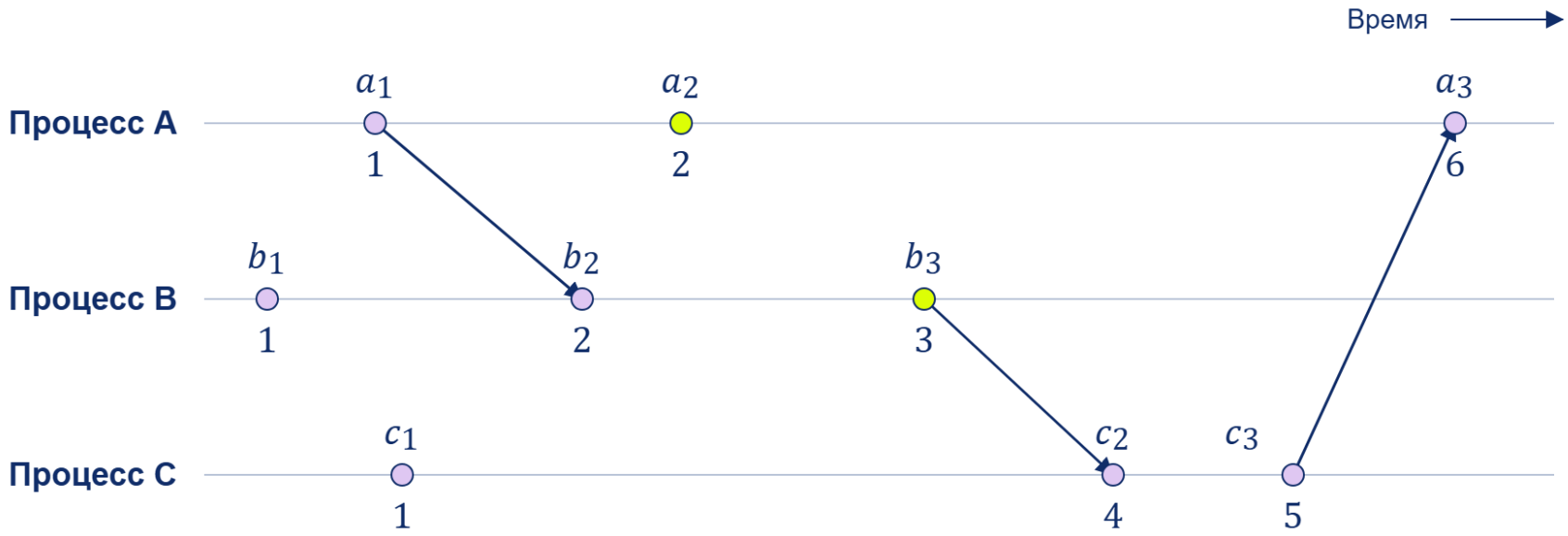
Изначально $L = 0$.

- Локальное событие или отправка: $L \leftarrow L + 1$.
- Сообщение несёт метку события отправки m.
- Получение сообщения с меткой m: $L \leftarrow \max(L, m) + 1$.

Меткой события служит значение L после обновления.



Свойства и ограничения часов Лэмпорта



Любые логические часы T должны сохранять причинный порядок:

$$a \to b \implies T(a) < T(b)$$

Обратная импликация для L неверна. Равные L не означают одновременности.

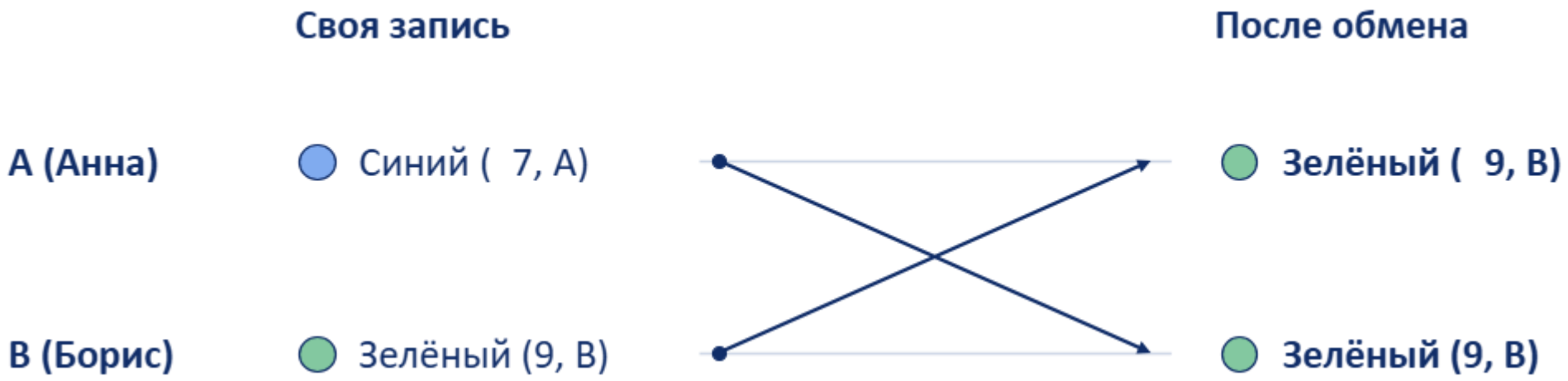
Пара $(L, PROC_ID)$ задаёт полный порядок, согласованный с причинностью.

Для последовательной согласованности нужен общий порядок, объясняющий все чтения.

Выбор версии по часам Лэмпорта

Анна и Борис независимо меняют цвет фона карточки товара.

По правилу LWW реплика сохраняет запись с наибольшей парой (счётчик, ID реплики).



Правило LWW даёт сходимость, но по меткам нельзя определить, видел ли Борис правку Анны.

Векторные часы

В метке события для каждого процесса записано, сколько его первых событий учтено.

Сравнение меток

$$V \leq W \Leftrightarrow \forall i: V[i] \leq W[i]$$

$$V < W \Leftrightarrow V \leq W \ \wedge \ V \neq W$$

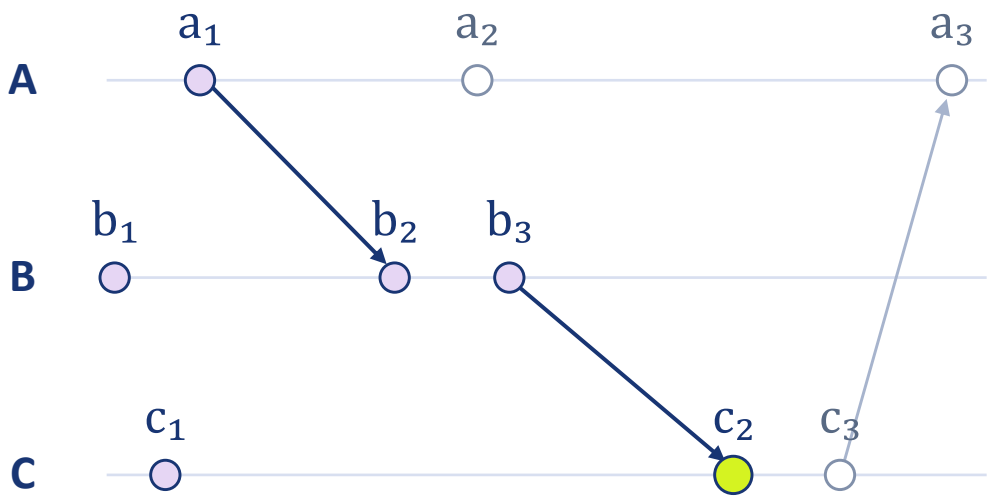
Причинный порядок

$$V(a) < V(b) \ \Leftrightarrow \ a \to b$$

Конкурентность

$$a \parallel b \ \Leftrightarrow \ V(a) \not\leq V(b) \ \wedge \ V(b) \not\leq V(a)$$

Какие события учтены в c_2?



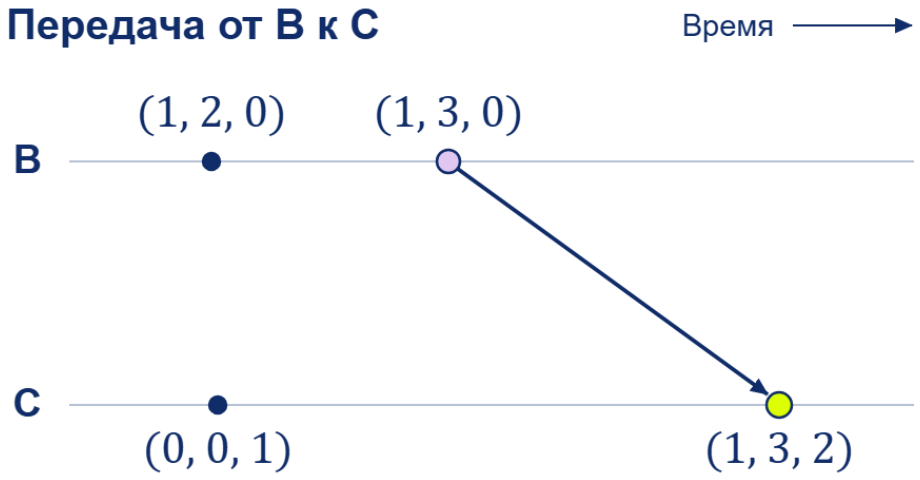
$$V(c_2) = (1, 3, 2)$$

Закрашены события, учтённые в c_2.

Как обновлять векторные часы

Изначально для трех процессов $V = (0,0,0)$. На каждом процессе i:

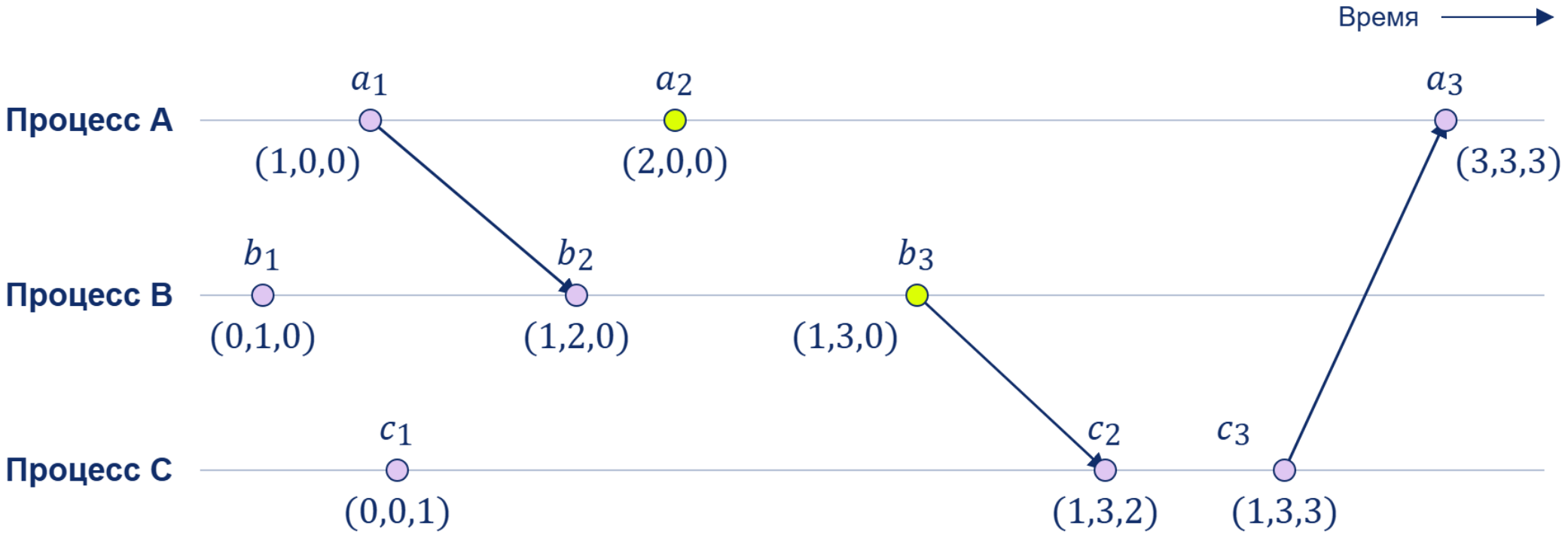
- Локальное событие или отправка: $V[i] \leftarrow V[i] + 1$.
- Сообщение несёт копию вектора после увеличения.
- Получение M: $V \leftarrow \max(V, M)$ покомпонентно, затем $V[i] \leftarrow V[i] + 1$.



Получение на C

	A	B	C
До получения	0	0	1
В сообщении	1	3	0
Максимум	1	3	1
			+1
Метка события	1	3	2

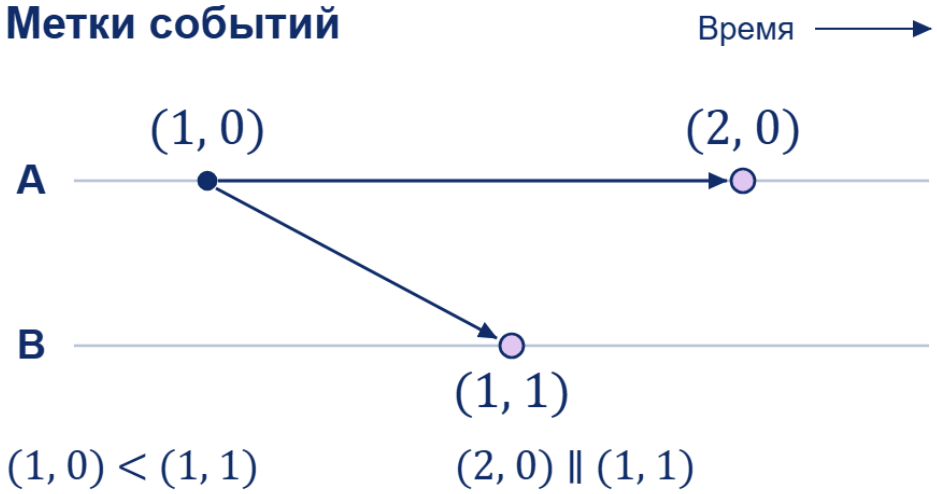
Векторные часы на той же диаграмме



Для a_1 и c_2 выполняется $V(a_1) < V(c_2)$. Для a_2 и b_3 их векторы несравнимы.

Результаты сравнения векторов

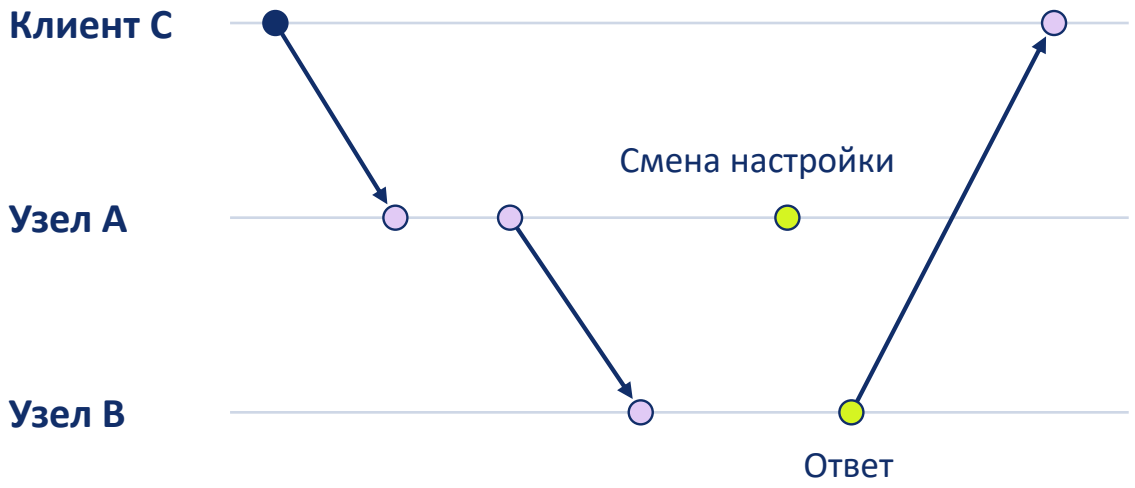
Векторы	Отношение	Для меток событий одного исполнения
(1,0) и (1,1)	Первый строго меньше	Первое событие предшествует второму
(2,0) и (1,1)	Несравнимы	События конкурентны
(2,1) и (2,1)	Равны	Одна метка одного события





Проверка зависимостей между событиями

Трасса выполнения системы

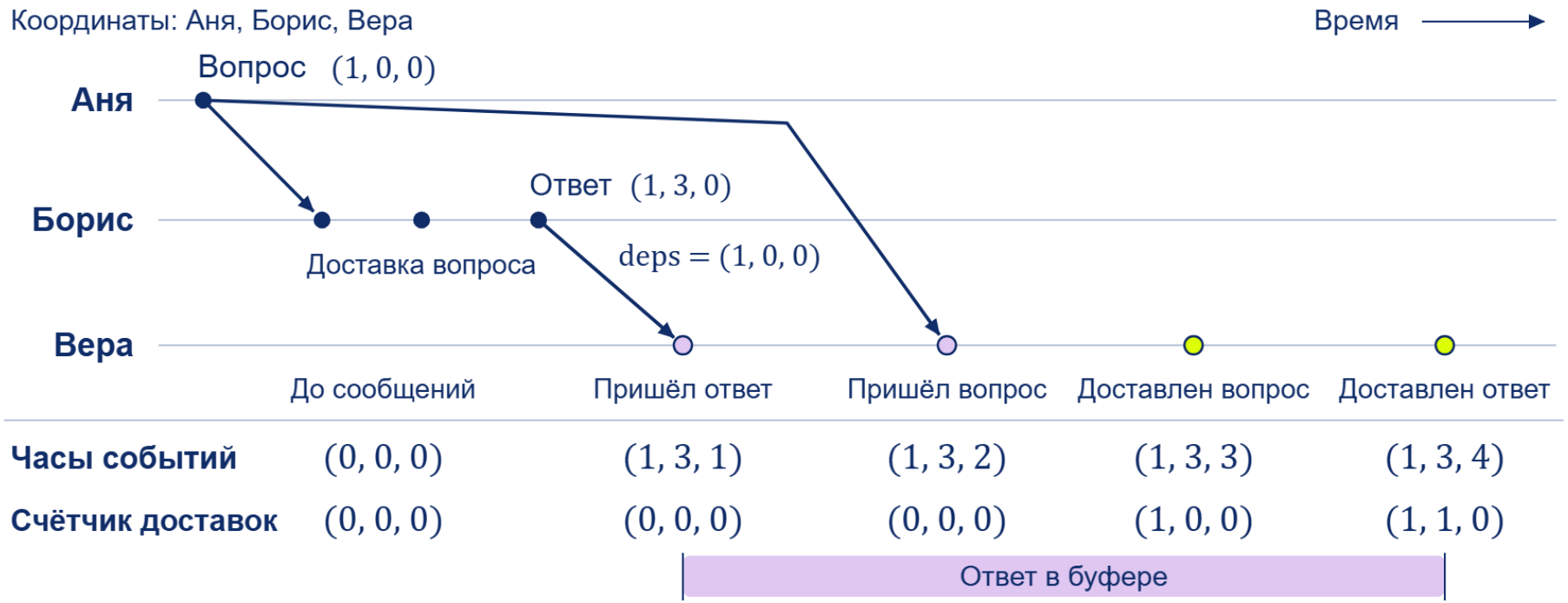


Общий журнал событий

Событие	(C, A, B)
A переслал запрос	$(1, 2, 0)$
A сменил настройку	$(1, 3, 0)$
B получил запрос	$(1, 2, 1)$
B отправил ответ	$(1, 2, 2)$

Смена настройки на A и отправка ответа на B конкурентны, то есть изменение настройки не могло повлиять на показанный ответ.

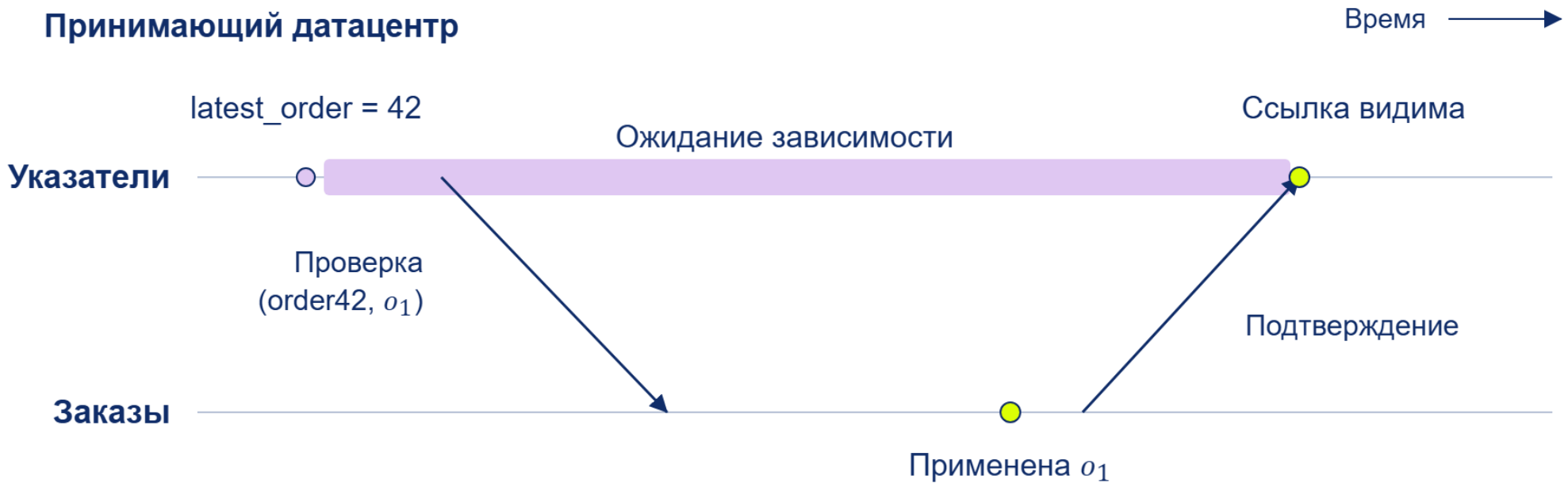
Векторы при причинной рассылке



Представление	Смысл координаты	Правило обновления
Часы событий	Префикс событий процесса в причинном прошлом вместе с текущим событием	На каждом своём событии +1; при получении перед этим выполняется max
Счётчик доставок	Число сообщений отправителя, доставленных приложению	+1 в координате отправителя только при доставке

Когда можно показать ссылку на заказ?

При причинной согласованности клиент, прочитавший ссылку, должен успешно прочитать и сам заказ.

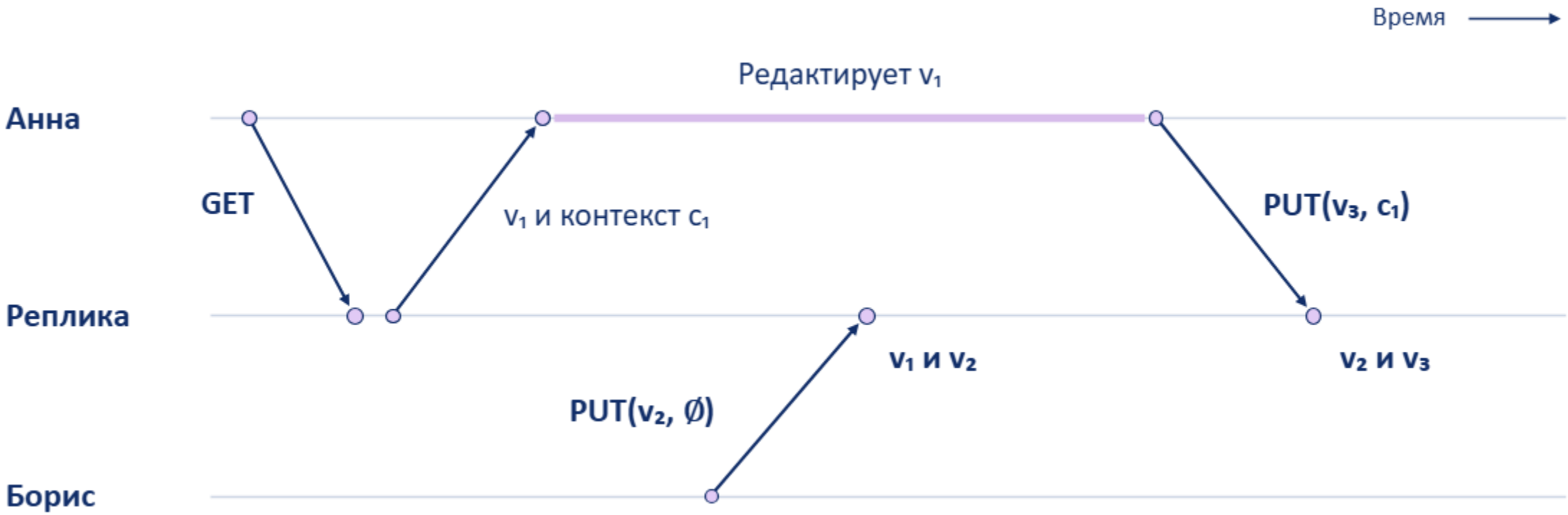


Вместе со ссылкой передаётся метка записи, от которой она зависит.

Шард указателей открывает ссылку для чтения после подтверждения от шарда заказов.

Какие версии заменяет PUT?

В начале на реплике одна редакция описания товара — v_1.



Контекст c_1 охватывает v_1. Реплика заменяет v_1 на v_3 и сохраняет правку Бориса v_2.

Потомок, старая версия и sibling

$H(v)$ — история версии v, включая её собственное изменение.

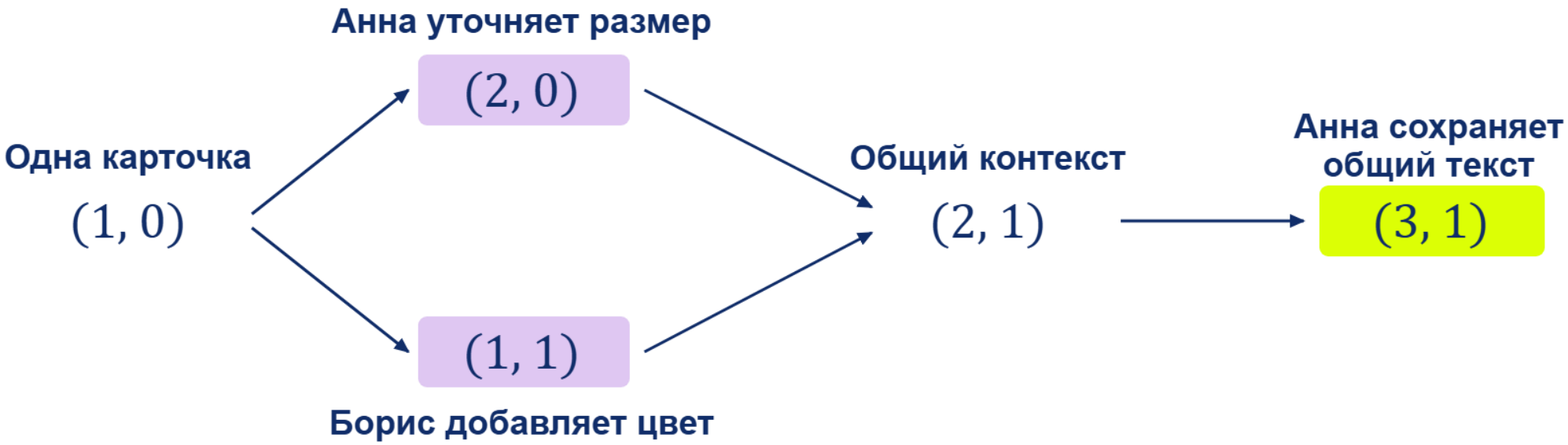
Сравниваем хранимую u и пришедшую v по их историям.

Случай	H(u)		H(v)	Действие реплики
Потомок	$\{a\}$	$\subsetneq$	$\{a, b\}$	Заменить u на v
Старая версия	$\{a, b\}$	$\supsetneq$	$\{a\}$	Оставить u
Повтор	$\{a, b\}$	$=$	$\{a, b\}$	Пропустить повтор
Конкурентные версии (siblings)	$\{a, b\}$	несравнимы	$\{a, c\}$	Сохранить u и v

Векторы версий

Вектор версий компактно представляет историю правок одного объекта.

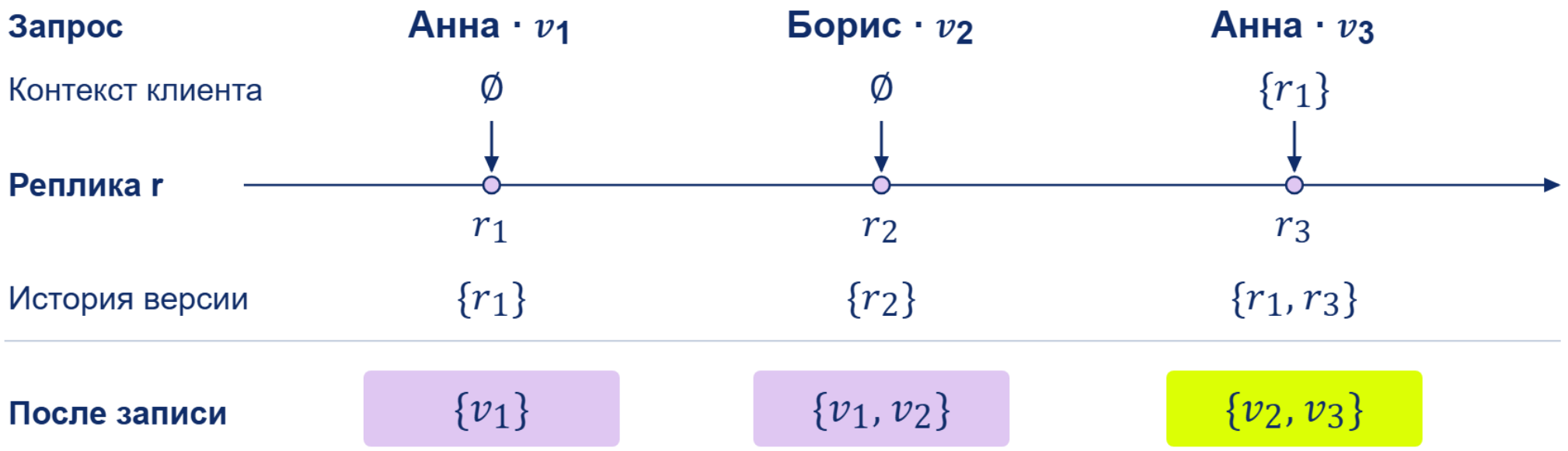
- У каждого редактора (клиента или реплики) своя компонента вектора.
- Значение k означает, что версия учитывает первые k правок этого редактора.



Что теряет вектор по репликам?

Теперь правки карточки нумерует реплика r, у Анны и Бориса один счётчик.

При повторном сохранении Анна передаёт контекст $\mathrm{c}_1 = \{\mathrm{r}_1\}$.



У версии Анны история $\{\mathrm{r}_1, \mathrm{r}_3\}$. Координата $(\mathrm{r}: 3)$ приписала бы ей и правку Бориса r_2.
Как сохранить обе редакции?

Что использовать для версии карточки?

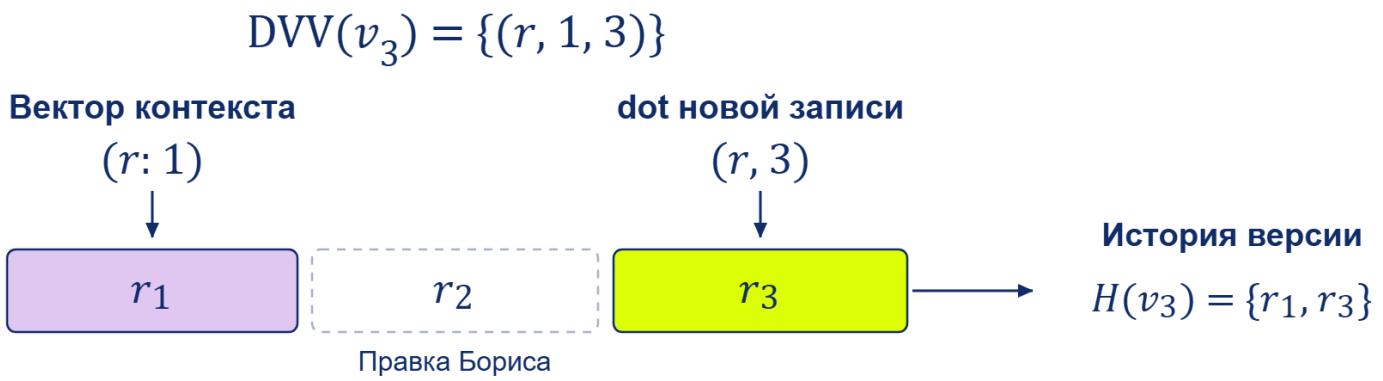
После трёх записей должны остаться правка Бориса v_2 и последняя правка Анны v_3.

Как задаём историю	Что получаем для v_2 и v_3	Что теряем или должны хранить
LWW по метке	Из меток 2 и 3 выбираем 3	Правка Бориса исчезает.
Вектор по клиентам	Борис (0, 1), Анна (2, 0). Версии несравнимы.	По компоненте на каждого редактора карточки.
Вектор по репликам	(r: 3) включает r_2.	Версия Анны ошибочно вытесняет версию Бориса.
Полная история	Борис $\{r_2\}$, Анна $\{r_1, r_3\}$. Версии несравнимы.	Храним идентификатор каждой учтённой правки.

Как оставить по координате на реплику и точно описать историю $\{r_1, r_3\}$?

Dotted version vector (DVV)

Добавим к вектору версий возможность явно указать отдельную правку (dot).



В компоненте (r, m, n) число m задаёт префикс до m-й правки реплики r, а пара (r, n) добавляет одну правку. При $m = 0$ префикс пуст.

Версия	Метка DVV целиком	История версии
v_1 — первая правка Анны	$\{(r, 0, 1)\}$	$\{r_1\}$
v_2 — правка Бориса	$\{(r, 0, 2)\}$	$\{r_2\}$
v_3 — вторая правка Анны	$\{(r, 1, 3)\}$	$\{r_1, r_3\}$

Когда между двумя PUT нужен GET?

Анна отправила PUT(v_1, ∅). Далее она продолжает редактировать тот же объект.



Если повторить PUT с пустым контекстом, обе правки Анны останутся конкурентными.

Для объединения нескольких версий клиенту нужны их содержимое и общий контекст.

Сколько dot может быть в DVV?

Каждая компонента DVV может содержать свой dot.
В истории $\{r_1, s_1, s_3, r_3\}$ по одному dot у каждой реплики.



Перед PUT через r Анна делает GET через s. Общий контекст прочитанных версий $(r\colon 1, s\colon 3)$ состоит из двух префиксов. Новая запись на r добавляет один dot $(r, 3)$.

Сравнение DVV

Для каждой реплики проверяем, входят ли в историю версии Y весь префикс и dot истории версии X.

- Да: версия X является устаревшей и её можно удалить.
- Нет: версии конкурентны и требуется хранить их до разрешения конфликта.

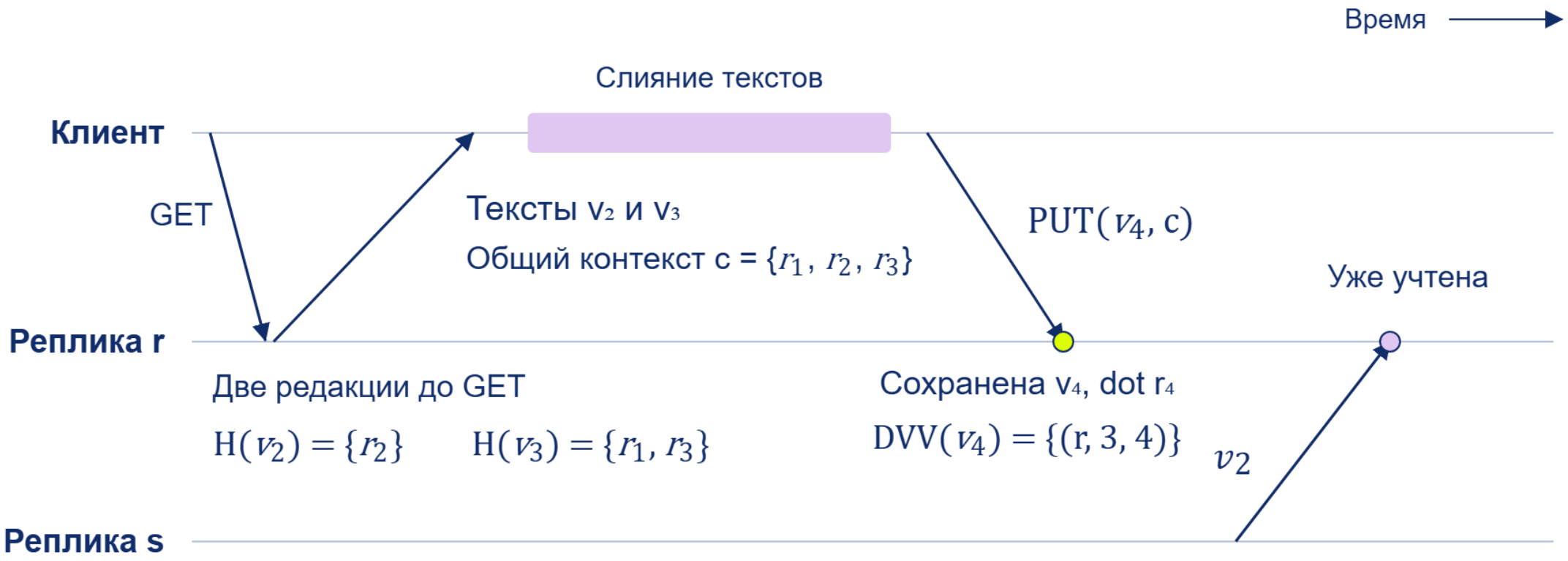
leq(X, Y) проверяет, включает ли история Y все правки из истории X.

m, p — длины префиксов. d, e — номера dot. Знак ∅ означает, что dot отсутствует.

```
leq(X, Y):
  для каждой реплики r из X:
    (m, d) = X[r]
    (p, e) = Y[r], по умолчанию (0, ∅)
    если не component_leq(m, d, p, e):
      вернуть ложь
  вернуть истину
```

```
component_leq(m, d, p, e):
  если d = ∅ и e = ∅: вернуть m ≤ p
  если d = ∅:
    вернуть m ≤ p или (m = e и e = p + 1)
  если e = ∅: вернуть d ≤ p
  вернуть d ≤ p или (d = e и m ≤ p)
```

Запись после разрешения конфликта



Как объединить две правки?

Редактор	Изменение	Метка записи
Анна, реплика A	**M → L**	(7, A)
Борис, реплика B	**синий → зелёный**	(9, B)

Один LWW-регистр на всю карточку

M, зелёный

Обе реплики выбрали карточку Бориса.

У каждого поля свой LWW-регистр

L, зелёный

Обе реплики сохранили размер Анны и цвет Бориса.

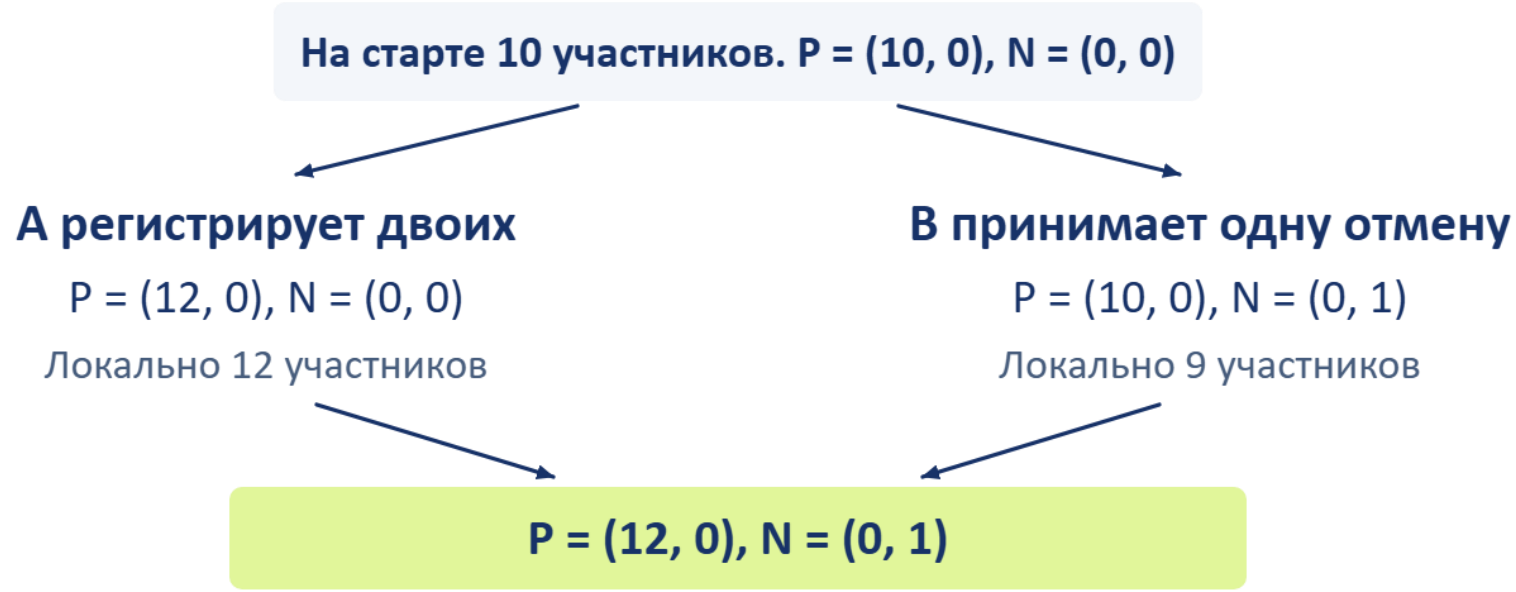
CRDT (*conflict-free replicated data type*) — реплицируемый тип данных, правила обновления, чтения и слияния которого гарантируют: учтя один и тот же набор обновлений, реплики получат одинаковое значение. Оба варианта дают CRDT.

CRDT для счётчика

PN-счётчик хранит два вектора. В P накапливаются увеличения, в N — уменьшения. Реплика i увеличивает только свои компоненты $P[i]$ и $N[i]$.

Чтение: $value = \Sigma_i P[i] - \Sigma_i N[i]$

Слияние: для каждого из двух векторов берётся покомпонентный максимум.



Обе реплики показывают 12 − 1 = 11. Повтор состояния B даёт те же 11.

Удаление товара из корзины

В **OR-set (*observed-remove set*)** каждый вызов ***add*** получает уникальную метку, даже если элемент уже есть в множестве.

Операция ***remove*** отменяет все добавления этого элемента, известные реплике в момент удаления. Элемент остаётся в множестве, пока у него есть хотя бы одна непогашенная метка.



Сходимость без sequential consistency



Ни один последовательный порядок операций, сохраняющий порядок действий каждого покупателя, не объясняет оба чтения.

Итоги лекции

- **Физическое время.** Погрешность при измерении времени и синхронизации часов неизбежна. Если интервалы неопределённости двух событий не пересекаются, известно, какое событие произошло раньше.
- **Логические часы.** Метки Лэмпорта растут вдоль причинных связей. Векторные часы позволяют также распознать конкурентные события.
- **Причинная согласованность.** Запись становится видимой клиенту только тогда, когда ему доступны и её зависимости. Клиенты могут наблюдать конкурентные изменения в разном порядке.
- **Версии данных.** Контекст PUT указывает, какие версии заменяет новая запись. Векторы версий позволяют выявлять конкурентные версии. DVV позволяет также учесть новую правку, сохранив пропуск в истории версии.
- **CRDT.** Реплики, учтя одни и те же обновления, дают одинаковый результат. Сходимость не гарантирует последовательной согласованности.

Материалы к лекции

Время и порядок событий

Distributed Systems · §8.1
Клеппман. Часы (гл. 3, §4.1) и слияние правок полей

NTPv4 · TrueTime · Spanner
Смещение часов в NTP и commit wait в Spanner

The Trouble with Timestamps
Как LWW теряет запись при откате часов

How to Do Distributed Locking
Истечение аренды и проверка fencing token

Часы Лэмпорта · Векторные часы
Как метки отражают причинный порядок событий

ShiVector и ShiViz · Dinv
Журналы событий и согласованные срезы

Согласованность и версии

Consistency Models · Session Guarantees
Модели согласованности и чтение своих записей

COPS
Проверка зависимостей между ключами, §4.3–4.5

Dynamo
Ветвление и слияние версий, §4.4, рис. 3

Dotted Version Vectors
Контекст GET/PUT и устройство DVV, §3–5

Vector Clocks Revisited, Part 2
Почему в Riak появляются лишние версии

High Availability under Eventual Consistency
Бакеро. PN-счётчик · Корзина с наблюдаемым удалением

Approaches to CRDTs · §3.2 · §3.3
Удаление в OR-Set и последовательная согласованность