10. Время, часы и порядок событий

Сухорослов Олег Викторович

13.11.2023

План лекции

- Физические часы
- Логические часы
  - Часы Лэмпорта
  - Векторные часы
- Применение логических часов
  - Упорядоченная рассылка
  - Обнаружение конфликтов при репликации данных

https://fatcatart.com/2012/05/postoyanstvo-pamyati

Зачем нужны часы в РС?

- Измерять длительность некоторого процесса во времени
  - Профилирование, таймауты, детекторы отказов, планировщики...
- Определять момент времени, когда произошло некоторое событие
  - Логирование событий, время покупки в базе данных...
- Проверять наступил ли некоторый момент времени
  - Проверка валидности цифрового сертификата...
- Определять порядок событий, происходящих на разных узлах РС

Типы часов

- Физические
  - измеряют число прошедших секунд
  - сложно использовать в РС в силу отличий (смещения и дрейфа) часов на узлах
- Логические
  - измеряют число произошедших событий (например, отправленных сообщений)
  - позволяют фиксировать причинно-следственные связи

Физические часы

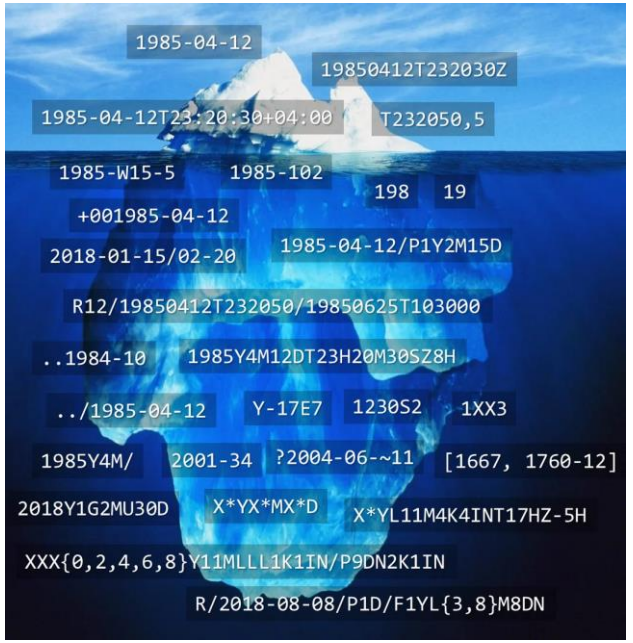
- Кварцевые часы
  - Частота колебаний кварцевого генератора известна с некоторой погрешностью
  - Типичный допустимый дрейф часов: 20-50 ppm (~10-25 минут в год)
  - Частота может существенно зависеть от температуры
- Атомные часы
  - 1 секунда = 9 192 631 770 периодов излучения, возникающих при переходах между уровнями состояния атома цезия-133
  - Точность 10^{-14} (1 секунда в 3 миллиона лет)
  - Цена значительно дороже
  - Используются в GPS

Стандарты времени

- Среднее время по Гринвичу (GMT)
  - основано на астрономических наблюдениях
  - скорость вращения Земли не является постоянной
- Международное атомное время (TAI)
  - основано на квантовой механике
- Всемирное координированное время (UTC)
  - TAI с корректировками, учитывающими вращение Земли
  - коррекции в форме leap second применяются 30 июня и 31 декабря

Представление времени в компьютере

- Unix time
  - число секунд с 00:00 1 января 1970 года UTC
  - нет учёта leap seconds
- ISO 8601
  - текстовый формат представления времени в UTC
  - `2020-11-21T16:30:56+03:00`
- Преобразование между форматами
  - Григорианский календарь
  - знание прошлых и будущих leap seconds
- Много ПО просто игнорирует leap seconds
  - но в ОС и РС часто требуется измерять время с высокой точностью



The ISO 8601 Iceberg

Инцидент 30 июня 2012 года

INTERNATIONAL EARTH ROTATION AND REFERENCE SYSTEMS SERVICE (IERS)

SERVICE INTERNATIONAL DE LA ROTATION TERRESTRE ET DES SYSTEMES DE REFERENCE

SERVICE DE LA ROTATION TERRESTRE
OBSERVATOIRE DE PARIS
61, Av. de l'Observatoire 75014 PARIS (France)
Tel. : 33 (0) 1 40 51 22 26
FAX : 33 (0) 1 40 51 22 91
e-mail : services.iers@obspm.fr
http://hpiers.obspm.fr/eop-pc

Paris, 5 January 2012

Bulletin C 43

To authorities responsible
for the measurement and
distribution of time

UTC TIME STEP
on the 1st of July 2012

A positive leap second will be introduced at the end of June 2012.
The sequence of dates of the UTC second markers will be:

2012 June 30, 23h 59m 59s
2012 June 30, 23h 59m 60s
2012 July 1, 0h 0m 0s

The difference between UTC and the International Atomic Time TAI is:

from 2009 January 1, 0h UTC, to 2012 July 1 0h UTC : UTC-TAI = - 34s
from 2012 July 1, 0h UTC, until further notice : UTC-TAI = - 35s

Leap seconds can be introduced in UTC at the end of the months of December or June, depending on the evolution of UT1-TAI. Bulletin C is mailed every six months, either to announce a time step in UTC or to confirm that there will be no time step at the next possible date.

Daniel GAMBIS
Head
Earth Orientation Center of IERS
Observatoire de Paris, France

Leap Second crashes half the internet

Yesterday's leap second killed half the Internet, including Pirate Bay, Reddit, LinkedIn, Gawker Media and a host of other sites. Even an airline. Any Linux user processes that depends on kernel threads had a high chance of failing. That includes MySQL and many Java servers like webapps, Hadoop, Cassandra, etc. The symptom was the user process spinning at 100% CPU even after being restarted. A quick fix seems to be setting the system clock which apparently resets the bad state in the kernel (we hope).

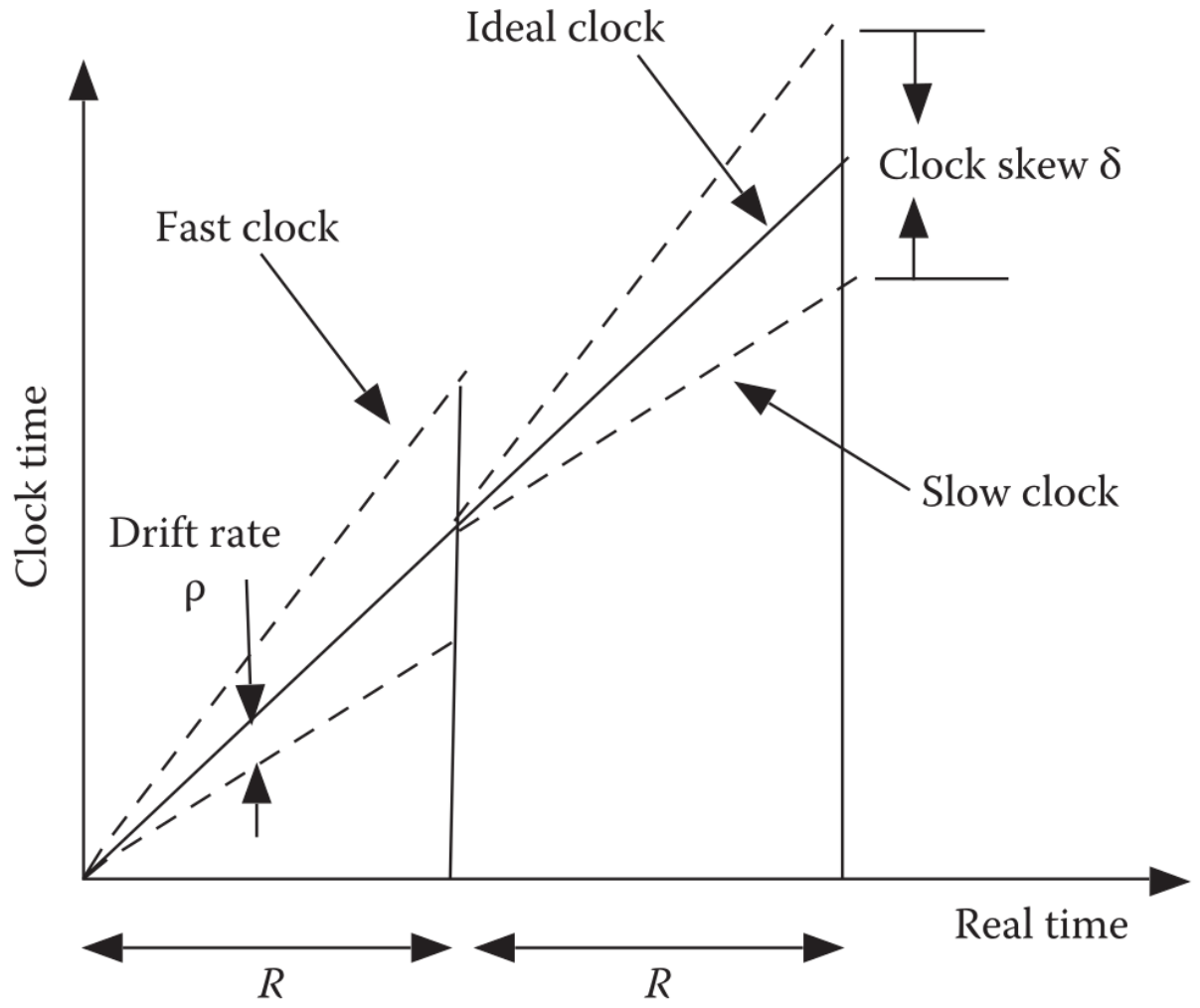
The underlying cause is something about how the kernel handled the extra second broke the futex locks used by threaded processes. Here's a very detailed analysis on the failing code but I'm not sure it's correct. According to this analysis the bug was introduced in 2008, then fixed in March 2012. But it may be the March fix is part of the problem. OTOH most of the systems that failed will be running kernels older than March so the problem must go further back. There's a kernel fix and also a detailed analysis. Time is hard, let's go shopping.

Leap Second crashes half the internet

The Inside Story of the Extra Second That Crashed the Web

Planes will crash! Things that leap seconds didn't, and did, cause

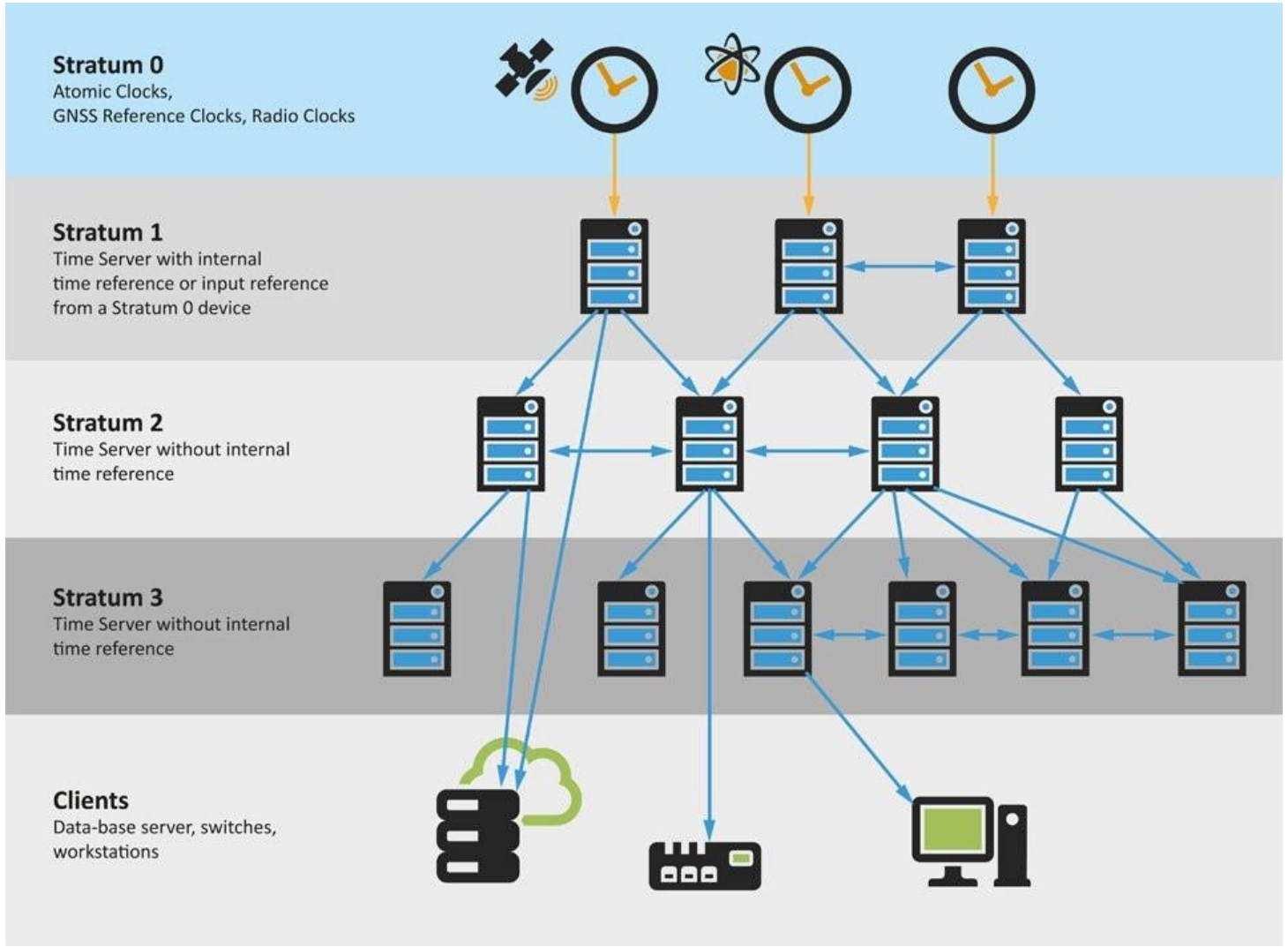
Смещение и дрейф часов



Синхронизация часов

- Цель: поддерживать смещение часов в заданных пределах
- Network Time Protocol (NTP): **~0.1-100 мс**
- Precision Time Protocol (PTP), Pulse Per Second (PPS), Data center Time Protocol (DTP): **~10-100 нс** (требуют аппаратной поддержки)
- Huygens: **~10 нс** (полностью программное решение)
- Google TrueTime: **~1 мс** (в масштабах Интернета)
- Cloud: Amazon Time Sync, Azure

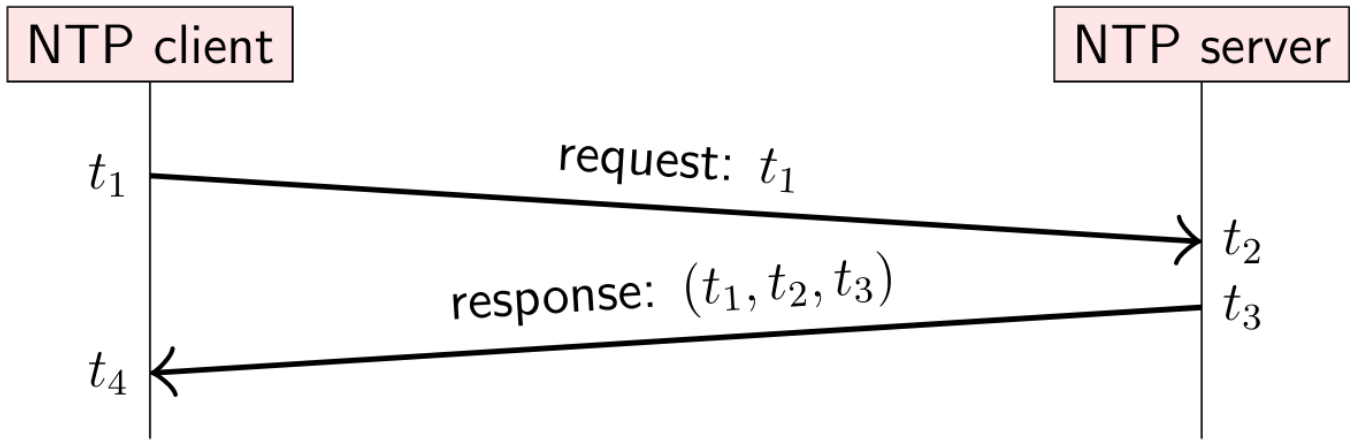
Network Time Protocol (NTP)



Network Time Protocol (NTP)

- Клиент-серверный протокол поверх UDP
- Иерархическая сеть серверов из нескольких слоев
  - Stratum 0: устройства с точными источниками времени (атомные часы, GPS-приемник...)
  - Stratum 1: серверы напрямую синхронизирующиеся с устройствами stratum 0
  - Stratum 2: серверы синхронизирующиеся по сети с серверами stratum 1 ...
- Клиент может взаимодействовать с несколькими серверами
  - Исключение неисправных серверов, усреднение измерений
- Клиент делает несколько запросов к одному серверу
  - Уменьшение ошибки, возникающей из-за изменений сетевой задержки

Оценка точного времени



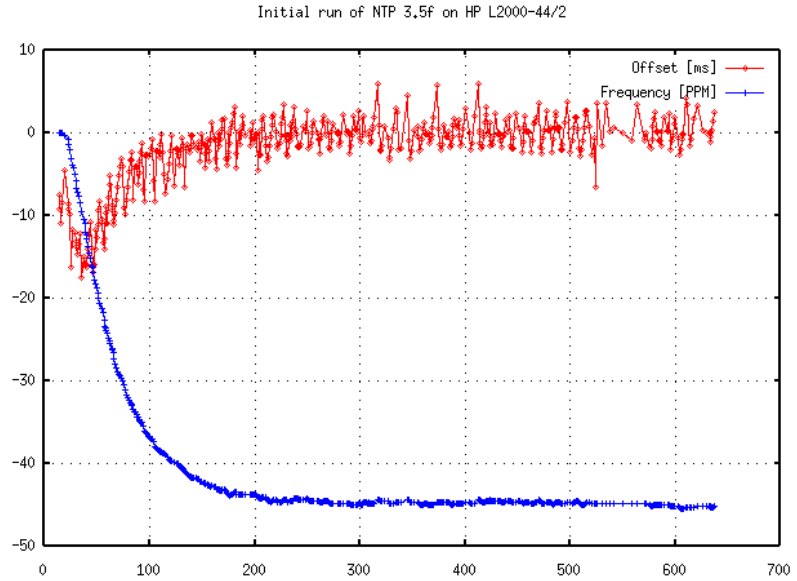
Round-trip network delay: $\delta = (t_4 - t_1) - (t_3 - t_2)$

Estimated server time when client receives response: $t_3 + \frac{\delta}{2}$

Estimated clock skew: $\theta = t_3 + \frac{\delta}{2} - t_4 = \frac{t_2 - t_1 + t_3 - t_4}{2}$

Коррекция часов клиента

- Смещение θ < 125 мс
  - небольшое замедление или ускорение часов
- Смещение 125 мс ≤ θ < 1000 с
  - сброс часов клиента на точное время
- Смещение θ ≥ 1000 с
  - ничего не делаем, оставляем решение проблемы за администратором
  - необходим мониторинг смещения часов



http://www.ntp.org/ntpfaq/NTP-s-algo/

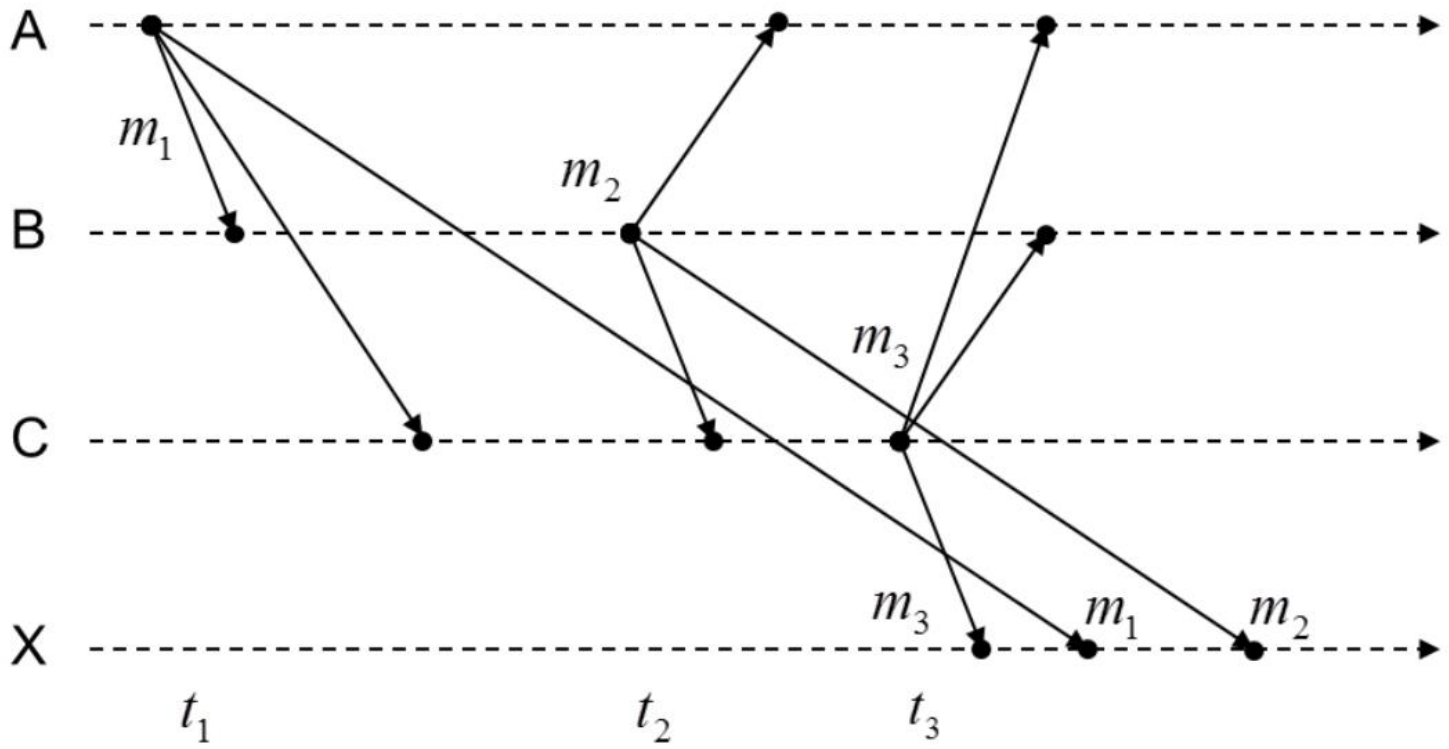
Локальные часы

- Time-of-day clocks
  - Привязаны к привычному (wall clock) времени
  - Могут "скакать" назад и вперед (см. NTP, leap seconds)
  - Не подходят для измерения интервалов времени
  - Linux: `clock_gettime(CLOCK_REALTIME)`
- Monotonic clocks
  - Гарантируется монотонное увеличение значения часов
  - Произвольная точка отсчёта (например, время загрузки машины)
  - Не подходит для сравнения времен между разными машинами
  - Обычно более высокое разрешение
  - Linux: `clock_gettime(CLOCK_MONOTONIC)`

Паузы в работе процессов

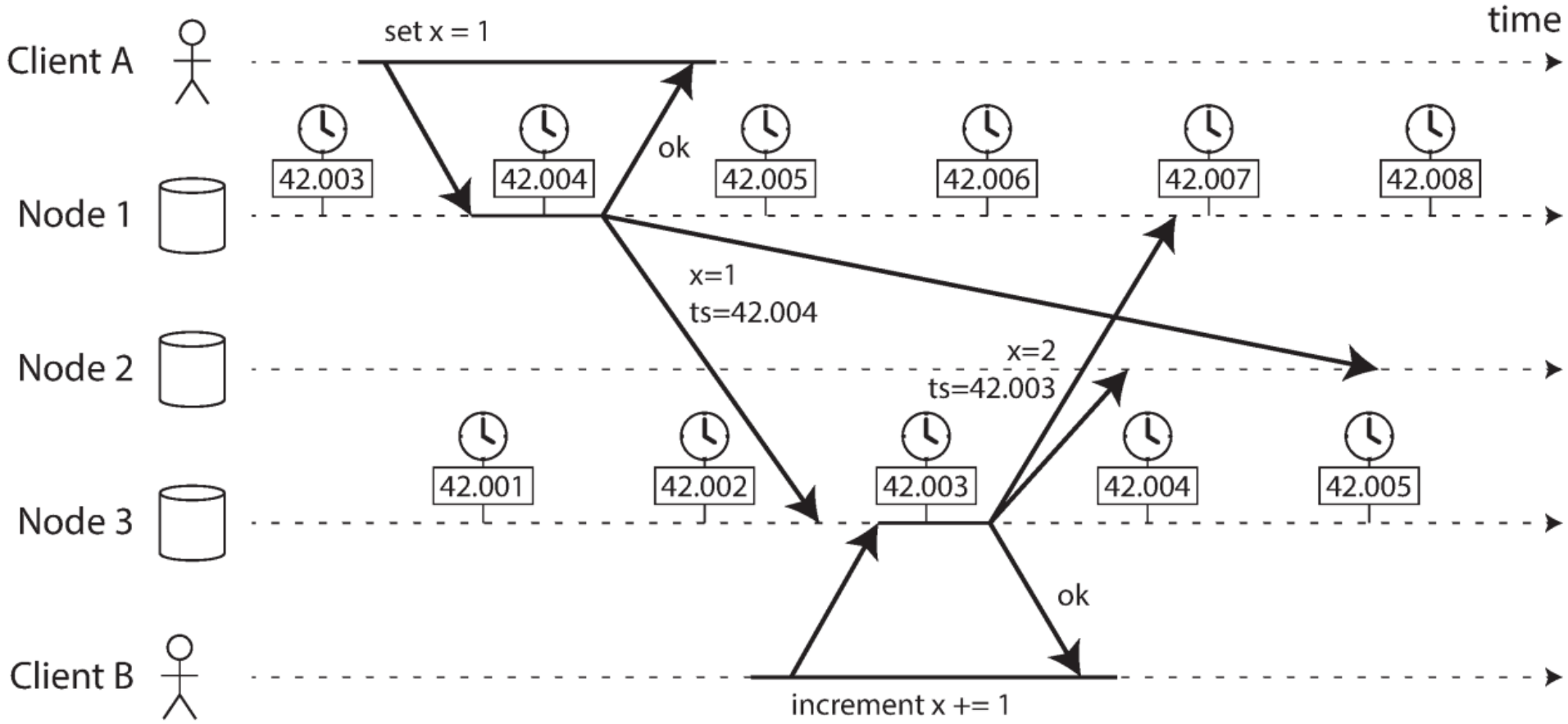
```
while (true) {
    request = getIncomingRequest();

    // Ensure that the lease always has at least 10 seconds remaining
    if (lease.expiryTimeMillis - System.currentTimeMillis() < 10000) {
        lease = lease.renew();
    }

    if (lease.isValid()) {
        process(request);
    }
}
```

Порядок сообщений

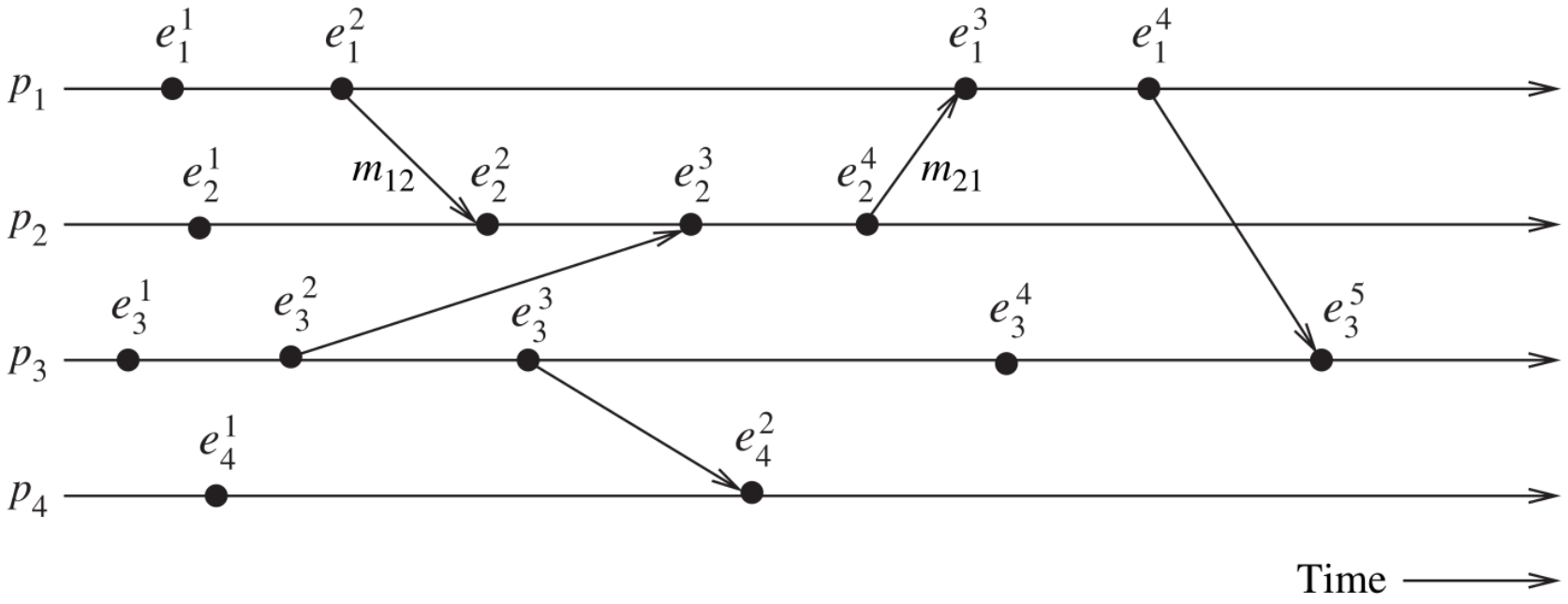


- Сообщение m_2 является ответом на m_1, а m_3 является ответом на m_2
- Процесс X получил сообщения в неправильном (с точки зрения логики) порядке
- Поможет ли упорядочить события физическое время?

Подход Last Write Wins

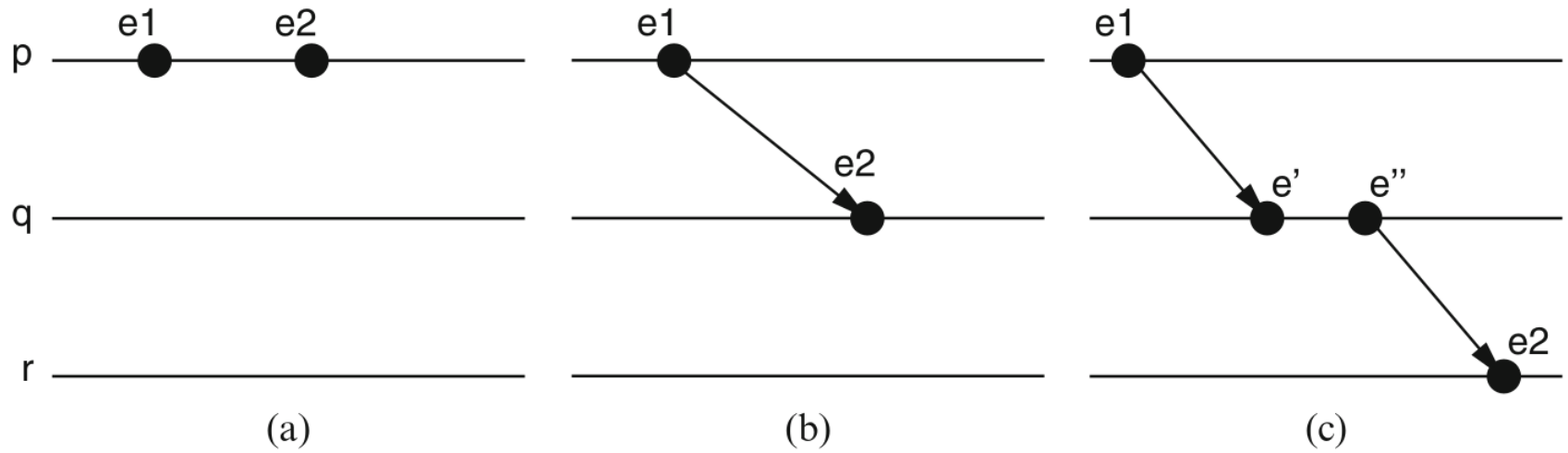


События в РС



- В каждом процессе происходят некоторые **события**: получение и отправка сообщений, шаг выполнения...
- Порядок событий в рамках процесса известен и соответствует локальным часам

Отношение happened-before

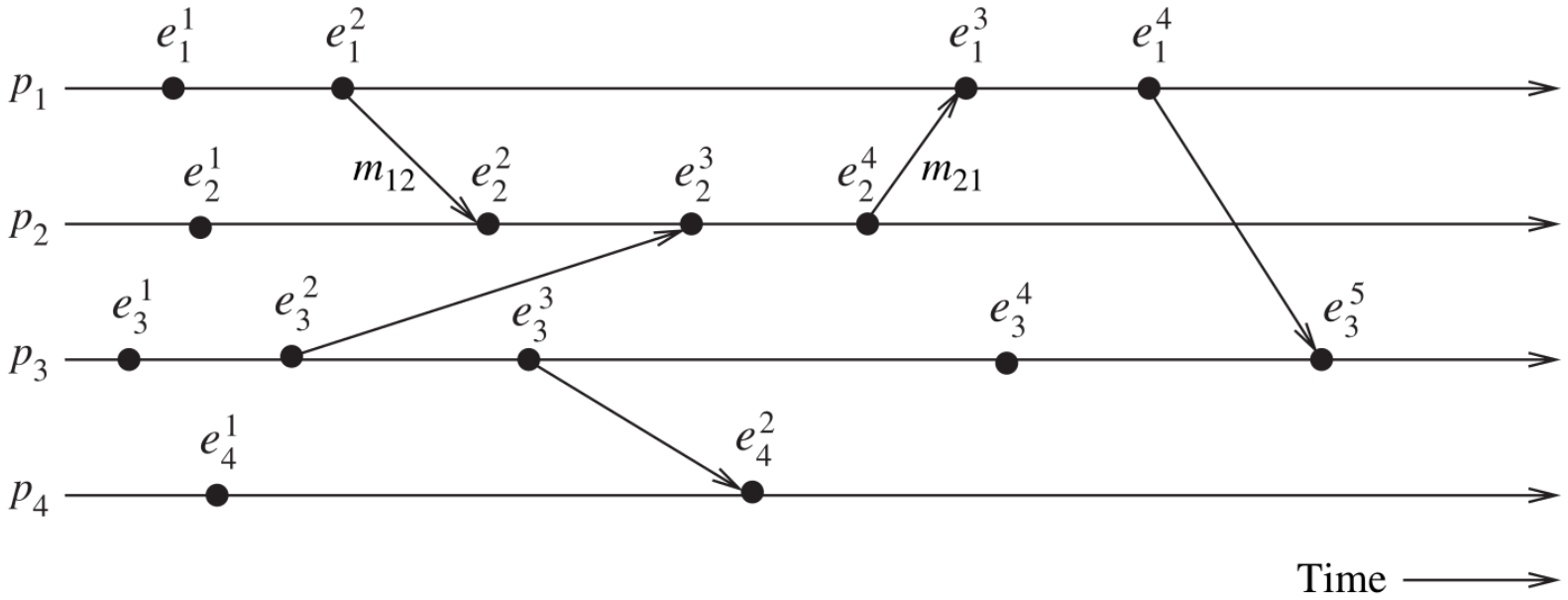


Событие a произошло до события b ($a \rightarrow b$) если выполняется одно из условий:

- a и b произошли на одном узле, и а произошло раньше
- a является отправкой сообщения m, а b является получением того же сообщения m
- существует событие c такое что $a \rightarrow c$ и $c \rightarrow b$

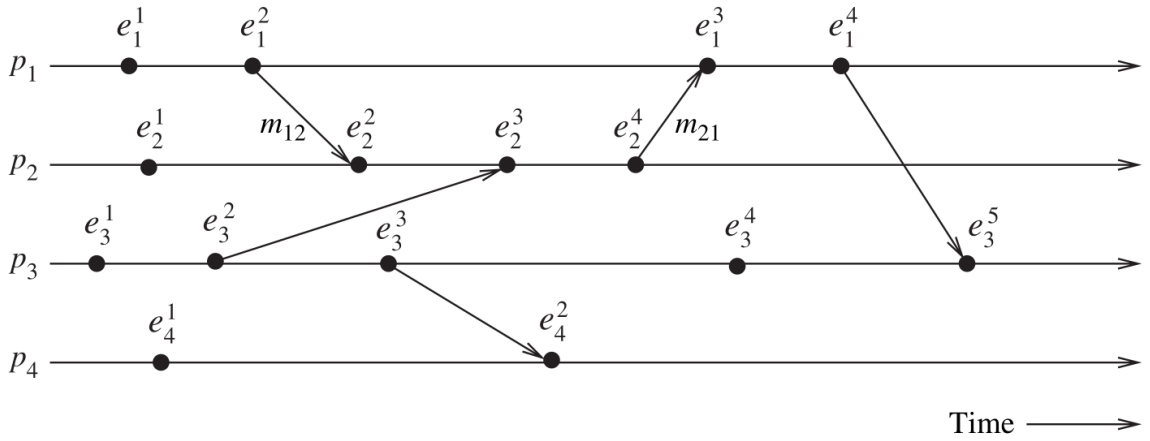
Отношение happened-before

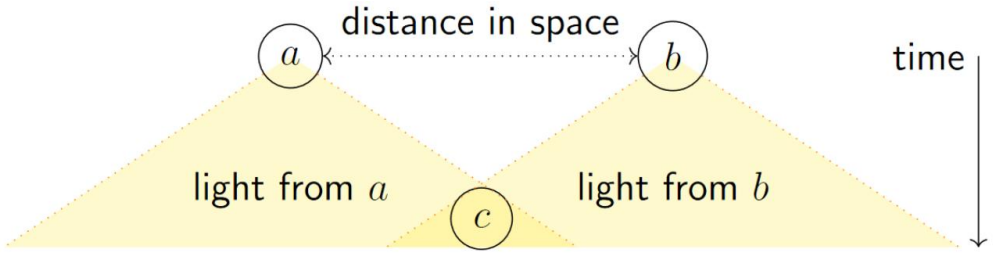
- Отношение happened-before определяет только частичный порядок
  - Возможно, что не выполняется ни $a \to b$, ни $b \to a$
  - Тогда события a и b называют одновременными ($a \parallel b$)
- Для любых событий a и b выполняется одно из трех: $a \to b$, $b \to a$, $a \parallel b$



Причинно-следственная связь

- Отношение happened-before указывает на возможную связь между событиями:
  - Если $a \rightarrow b$, то a могло повлиять на (быть причиной) b
  - Если $a \parallel b$, то a не могло повлиять на b

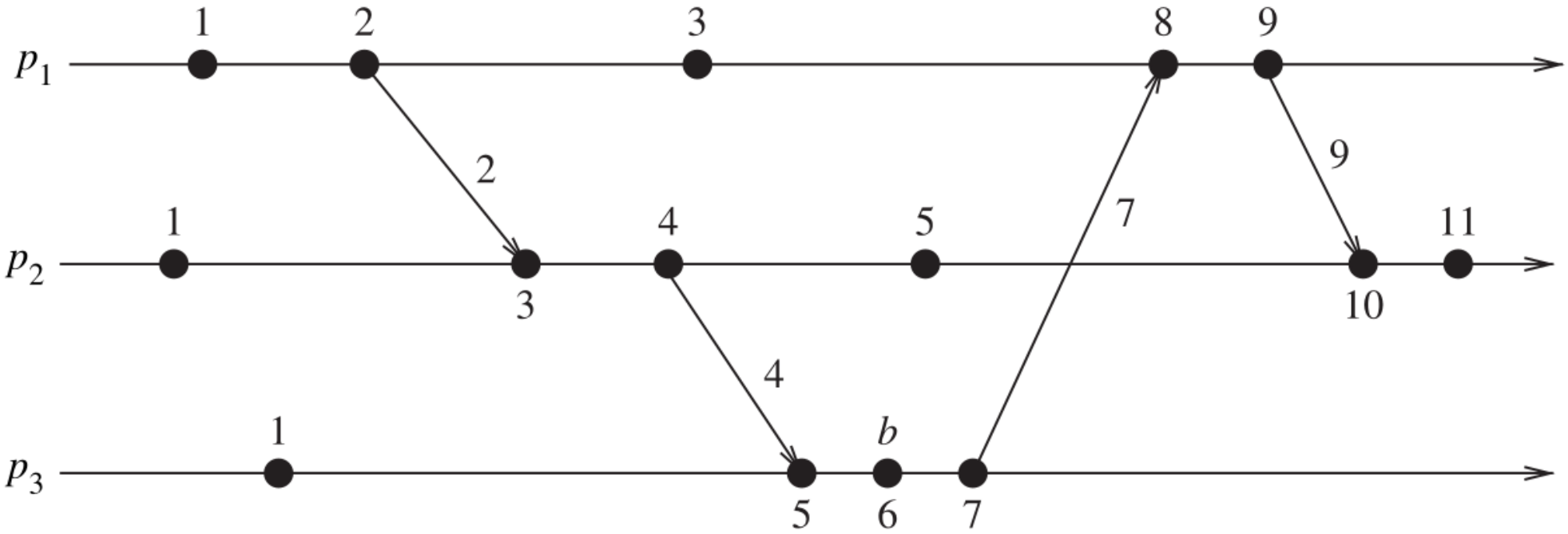




Логические часы

- Порядок $\prec$ является **причинным порядком**, если $a \rightarrow b \implies a \prec b$
- Физические часы
  - измеряют число прошедших секунд
  - сложно использовать в РС для получения причинного порядка событий
- Логические часы
  - измеряют число произошедших событий (например, отправленных сообщений)
  - не применимы для измерений моментов времени или длительности
  - сохраняют причинный порядок событий: $a \rightarrow b \implies T(a) < T(b)$

Часы Лэмпорта



Lamport L. Time, Clocks and the Ordering of Events in a Distributed System (1978)

Часы Лэмпорта: алгоритм

on initialisation **do**
    $t := 0$ ▷ each node has its own local variable t
end on

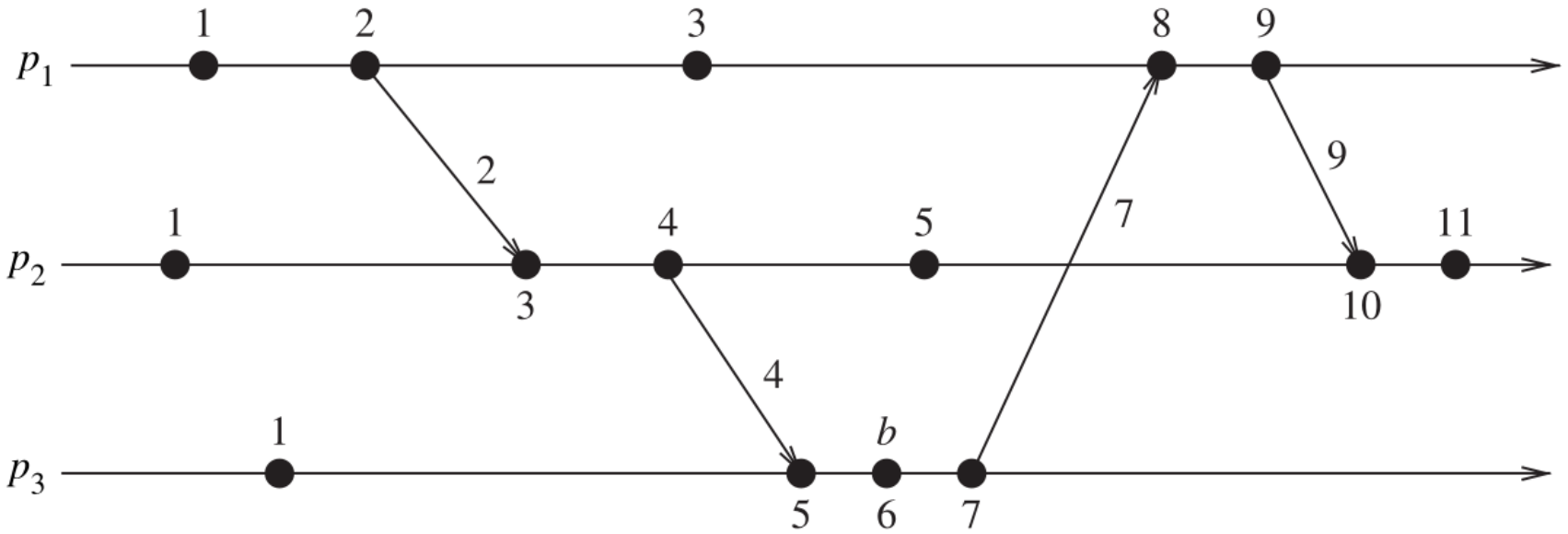
on any event occurring at the local node **do**
    $t := t + 1$
end on

on request to send message m **do**
    $t := t + 1$; send (t, m) via the underlying network link
end on

on receiving (t', m) via the underlying network link **do**
    $t := \max(t, t') + 1$
    deliver m to the application
end on

Часы Лэмпорта: свойства

- Если $a \to b$, то $L(a) < L(b)$
- Из $L(a) < L(b)$ не следует $a \to b$ (может быть $a \parallel b$)
- Возможно $L(a) = L(b)$ для $a \neq b$



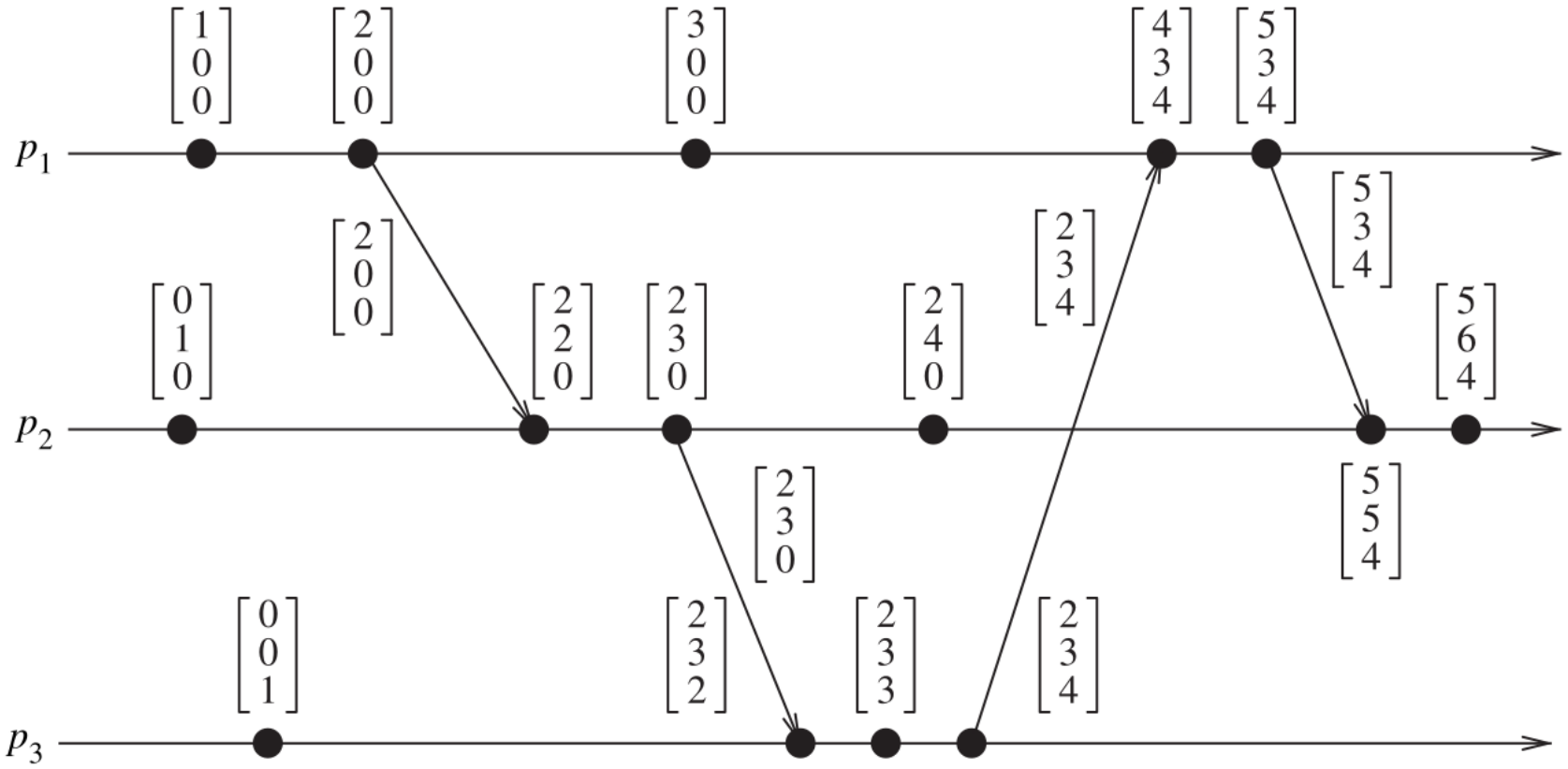
Порядок событий

- Пусть $P(e)$ - уникальный идентификатор процесса, на котором произошло событие e
- Пара $(L(e), P(e))$ уникально идентифицирует событие e
- Используя часы Лэмпорта можно определить линейный (полный) порядок событий:

$$a \prec b \iff L(a) < L(b) \vee (L(a) = L(b) \wedge P(a) < P(b))$$

- Этот порядок является причинным, то есть $a \to b \implies a \prec b$

Векторные часы



Fidge C.J. Timestamps in Message-Passing Systems That Preserve the Partial Ordering (1988)

Mattern F. Virtual Time and Global States of Distributed Systems (1988)

Векторные часы: алгоритм

on initialisation at node N_i **do**
    $T := \langle 0, 0, \ldots, 0 \rangle$ ▷ local variable at node N_i
end on

on any event occurring at node N_i **do**
    $T[i] := T[i] + 1$
end on

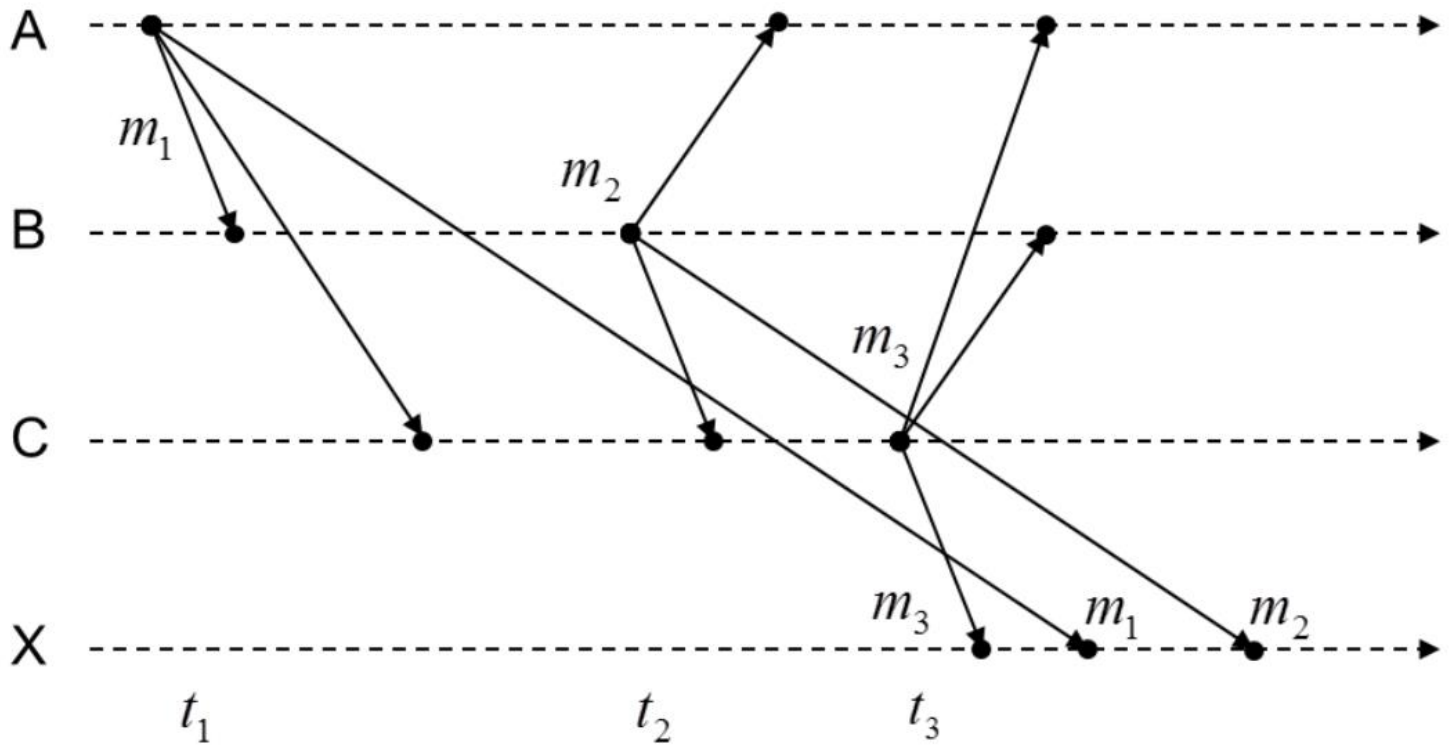
on request to send message m at node N_i **do**
    $T[i] := T[i] + 1$; send (T, m) via network
end on

on receiving (T', m) at node N_i via the network **do**
    $T[j] := \max(T[j], T'[j])$ for every $j \in \{1, \ldots, n\}$
    $T[i] := T[i] + 1$; deliver m to the application
end on

Векторные часы: свойства

- Правила сравнения значений векторного времени (n - число процессов)
  - $V = V' \iff V[i] = V'[i] \quad \forall\, i \in \{1, \ldots, n\}$
  - $V \leq V' \iff V[i] \leq V'[i] \quad \forall\, i \in \{1, \ldots, n\}$
  - $V < V' \iff V \leq V' \wedge V \neq V'$
  - $V \parallel V' \iff V \not\leq V' \wedge V' \not\leq V$
- Часы $V(a)$ кодируют набор событий, включая a и его зависимости $e \rightarrow a$
  - $V(a) \leq V(b) \iff (\{a\} \cup \{e | e \rightarrow a\}) \subseteq (\{b\} \cup \{e | e \rightarrow b\})$
- Это приводит к следующим свойствам векторных часов
  - $V(a) < V(b) \iff a \rightarrow b$
  - $V(a) = V(b) \iff a = b$
  - $V(a) \parallel V(b) \iff a \parallel b$

Порядок сообщений



- Сообщение m_2 является ответом на m_1, а m_3 является ответом на m_2
- Процесс X получил сообщения в неправильном (с точки зрения логики) порядке
- Поможет ли упорядочить события логическое время?

Causal Broadcast

- Используется вариант векторных часов, см. лекцию 4
- Элемент $deps[i]$ на узле j содержит число сообщений от i, которые были доставлены на j
- $deps \leq delivered$ означает, что узел уже доставил все сообщения, которые должны предшествовать данному

on initialisation **do**
$sendSeq := 0;\ delivered := \langle 0, 0, \ldots, 0 \rangle;\ buffer := \{\}$
end on

on request to broadcast m at node N_i **do**
$deps := delivered; deps[i] := sendSeq$
send $(i, deps, m)$ via reliable broadcast
$sendSeq := sendSeq + 1$
end on

on receiving msg from reliable broadcast at node N_i **do**
$buffer := buffer \cup \{msg\}$
while $\exists (sender, deps, m) \in buffer.\ deps \leq delivered$ **do**
deliver m to the application
$buffer := buffer \setminus \{(sender, deps, m)\}$
$delivered[sender] := delivered[sender] + 1$
end while
end on

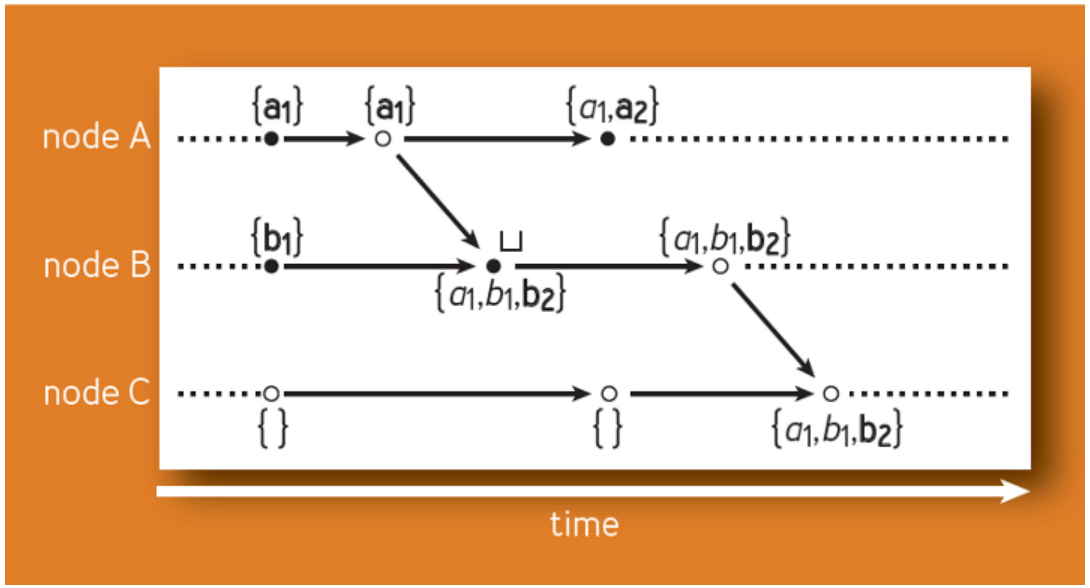
Total Order Broadcast

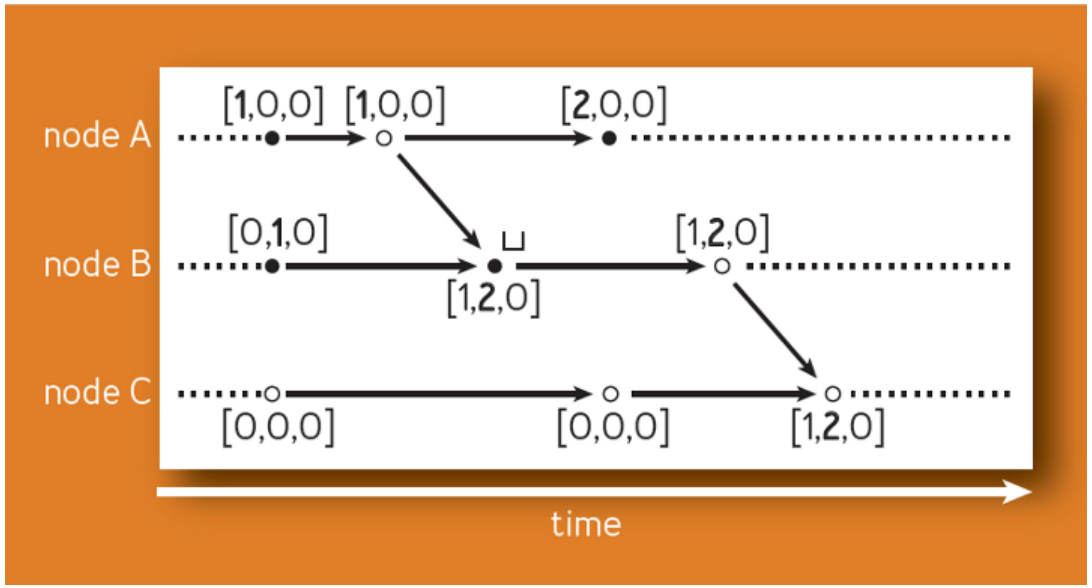
- Все процессы должны получить рассылаемые сообщения в одном порядке
- Реализация на основе **часов Лэмпорта**
  - Добавим к каждому рассылаемому сообщению значение часов
  - Будем доставлять сообщения в ранее описанном линейном порядке на основе значений часов и идентификаторов узлов
- Как процессу узнать, что он уже получил все сообщения с временем $\leq T$?
  - Если сообщения приходят в порядке их отправки (FIFO link), то надо дождаться сообщения с временем $> T$ от **каждого** процесса
  - Для этого можно использовать подтверждения или новые рассылки
- Проблема: отказ любого процесса блокирует доставку сообщений

Обнаружение конфликтов при репликации

- Операции записи могут приходить на реплики в разном порядке и дублироваться
- Надо уметь определять, когда пришедшее значение B **логически связано** с текущим хранимым значением A, а когда нет
  - Если запись B **произошла до** записи A, то значение A уже содержит эти изменения, и их можно игнорировать
  - Если запись A **произошла до** записи B, то значение B является более новой версий, и надо сохранить его вместо A
  - Если A и B не связаны логически (произошли **одновременно**), то возник **конфликт**, и надо сохранить оба значения до разрешения конфликта

Векторы версий (version vectors)



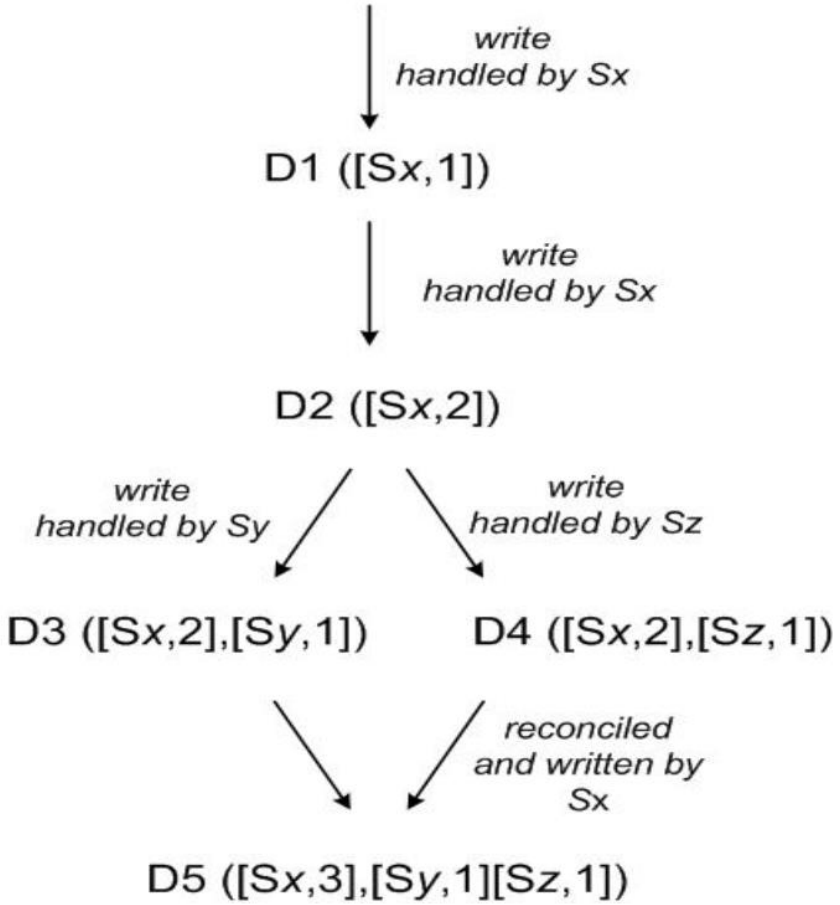


Parker D. Detection of Mutual Inconsistency in Distributed Systems (1983)

Векторы версий (version vectors)

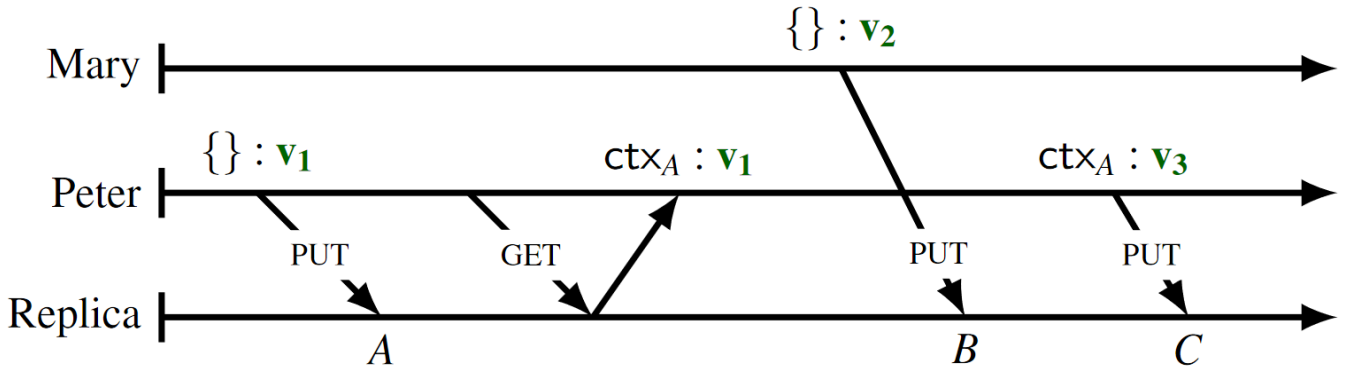
- Механизм, используемый для отслеживания логического порядка между версиями реплицируемых данных
- Подход аналогичен векторным часам
  - размер вектора равен числу реплик, изначально все значения равны 0
  - элемент $V[i]$ соответствует числу изменений, выполненных на реплике i
- Обновления векторов
  - при изменении данных на реплике i значение $V[i]$ увеличивается на 1
  - при получении изменения от другой реплики, векторы объединяются (максимум)
- Отслеживание порядка и конфликтов
  - если $V_{local} < V_{update}$, то $local \rightarrow update$ и изменение применяется
  - если $V_{local} \parallel V_{update}$, то возник конфликт (после его разрешения надо увеличить $V[i]$ на 1)
  - в противном случае это изменение уже есть, и его можно игнорировать

Применение векторов версий в Dynamo



Dynamo: Amazon's Highly Available Key-value Store (2007)

Конкурентные изменения к одной реплике



	lww	ch	vv$_{client}$	vv$_{server}$	dvv
A	17h00 : v1	$\{r_1\}$: v1	$\{(p,1)\}$: v1	$\{(r,1)\}$: {v1}	$((r,1),\{\})$: v1
ctx$_A$	$\{\}$	$\{r_1\}$	$\{(p,1)\}$	$\{(r,1)\}$	$\{(r,1)\}$
B	17h03 : v2	$\{r_1\}$: v1 $\{r_2\}$: v2	$\{(p,1)\}$: v1 $\{(m,1)\}$: v2	$\{(r,2)\}$: {v1, v2}	$((r,1),\{\})$: v1 $((r,2),\{\})$: v2
C	17h07 : v3	$\{r_2\}$: v2 $\{r_1,r_3\}$: v3	$\{(m,1)\}$: v2 $\{(p,2)\}$: v3	$\{(r,3)\}$: {v1, v2, v3}	$((r,2),\{\})$: v2 $((r,3),\{(r,1)\})$: v3

- Векторы версий не позволяют эффективно отслеживать изменения такого типа
- Dotted Version Vectors (2010) устраняют этот недостаток путем хранения пар (dot, vv)

Автоматическое разрешение конфликтов

- Специфичный для приложения метод
  - См. слияние версий корзин покупок в Dynamo
- Conflict-free Replicated Data Types (CRDTs)
  - Operation-based
  - State-based
- Operational Transformation (OT)

См. Kleppmann M. Distributed Systems (video, notes)

Материалы

- Distributed Systems Course (часть 3 и раздел 4.1)
- Distributed Systems: Principles and Paradigms (разделы 5.1-5.2)
- There is No Now: Problems with simultaneity in distributed systems
- Patterns of Distributed Systems: Version Vector
- Call Me Maybe: Cassandra
- The trouble with timestamps
- TrueTime
- Living Without Atomic Clocks
- Hybrid Logical Clocks
- Упомянутые статьи