



Time In Distributed Systems

Pt.1

Hello



Denis Golovachev



Grid Dynamics



Software Architect



Saint-Petersburg



вэлосити

брифинг

кипай

резать кости

рисерч

продакт

инсайт

кофаундер

диджитал-медиа

факап

фандрэйзинг

фокус-группа

дэдлайн

ключевой вендор

эккаунт-планирование

деманды

чек поинт

вижн

бэкграунд

кейс

фидбэк

челлендж

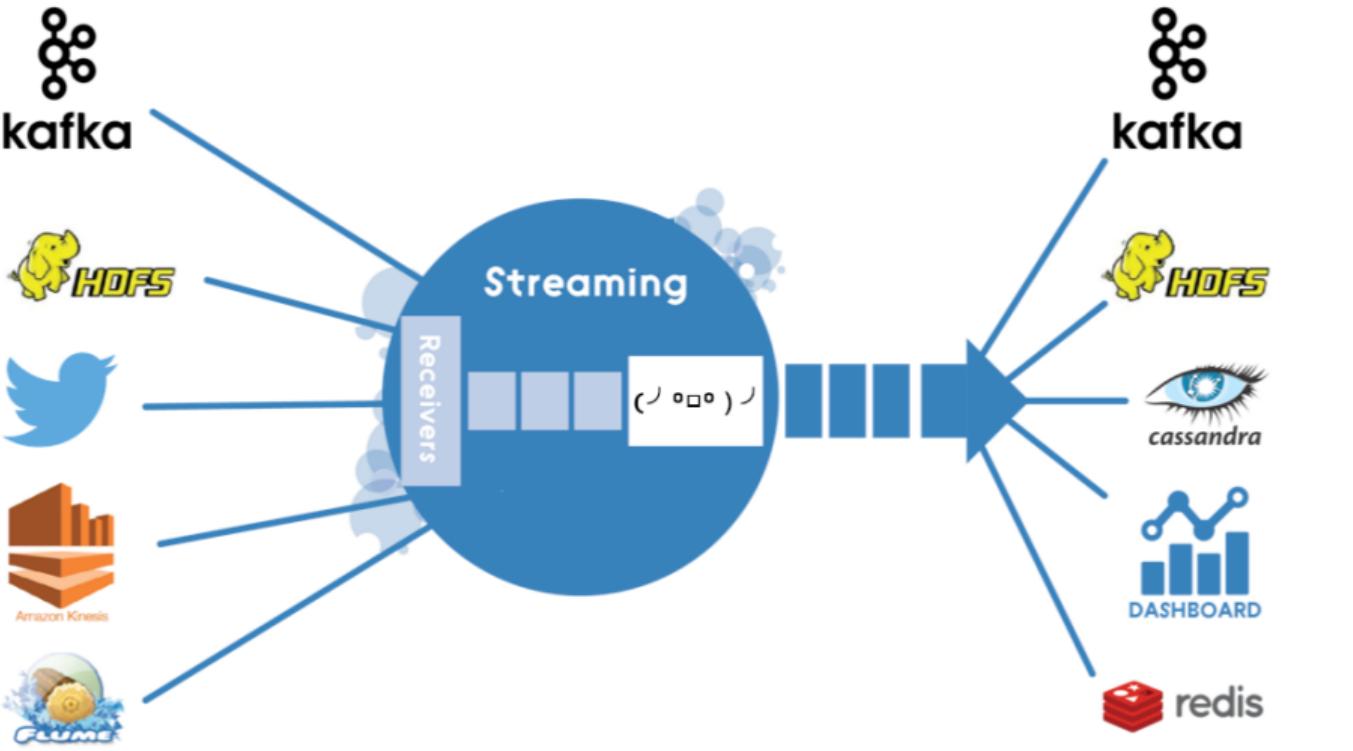
вау-эффект

- Common Clock
 - Causality
 - Fiscal year
- Drift and Skew
 - Peer to Peer
 - Exactly-once delivery

Но вернемся к нашему приветствию



Последние несколько лет я занимаюсь Стиминг Биг
дата решениями
Схематически наши системы можно представить как-
то так





И наш стек плюс-минус состоит из этих технологий
Так что если у вас есть желание поговорить о них, я
готов с радостью это сделать после выступления



Так же у нас есть биг data школа
в которой я являюсь профессором
И может показаться что лекция от профессора
должна быть



Professor

Так же у нас есть биг data школа
в которой я являюсь профессором
И может показаться что лекция от профессора
должна быть

Boring

Но я попробую вас в этом разубедить



Grid Dynamics

Ну что же, поехали. биг дата, грид динамикс, время

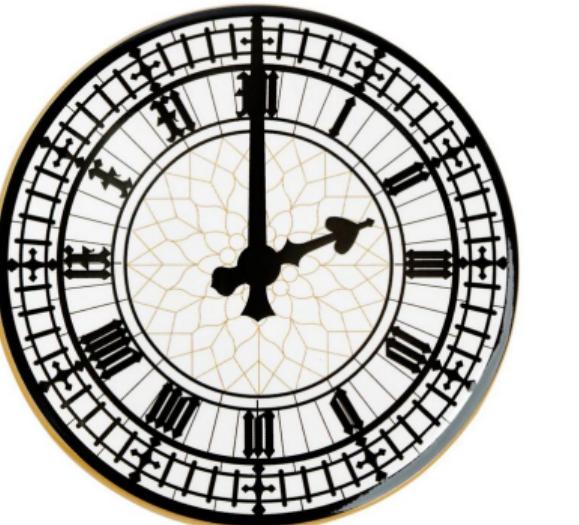
Time



И сразу хочу сказать что время, тема обширная и мы можем говорить о ней в том или другом контексте поэтому сразу давайте сузим область. Сегодня мы будем говорить о

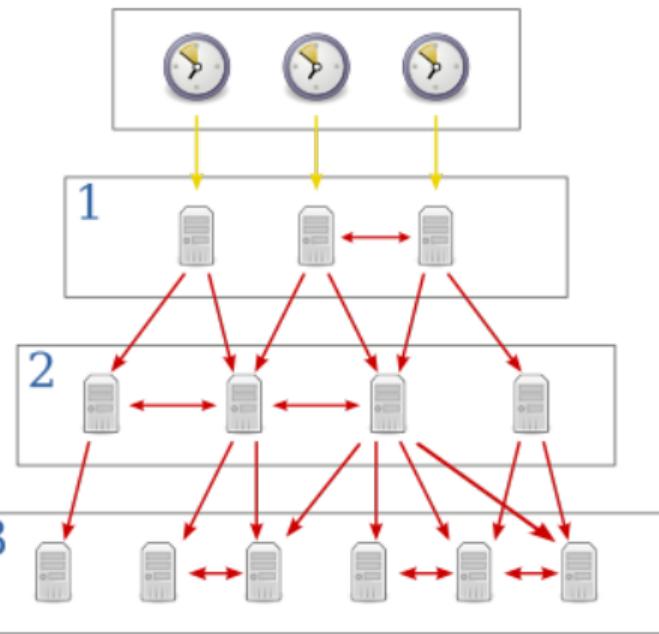
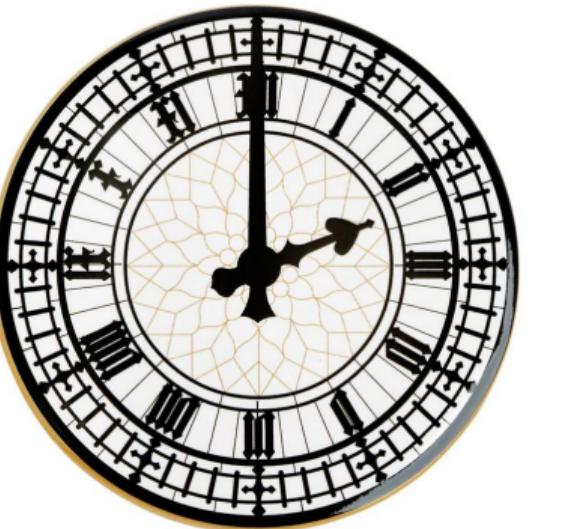


И сразу хочу сказать что время, тема обширная и мы можем говорить о ней в том или другом контексте поэтому сразу давайте сузим область. Сегодня мы будем говорить о



А о часах, календарях, таймерах и о том как мы измеряем время, особенно в распределенных системах

Time



А о часах, календарях, таймерах и о том как мы измеряем время, особенно в распределенных системах

What is time?

Начнем с вопроса
что такое время? К примеру я бы мог ответить

Time is ...



Way of arrange the order of events

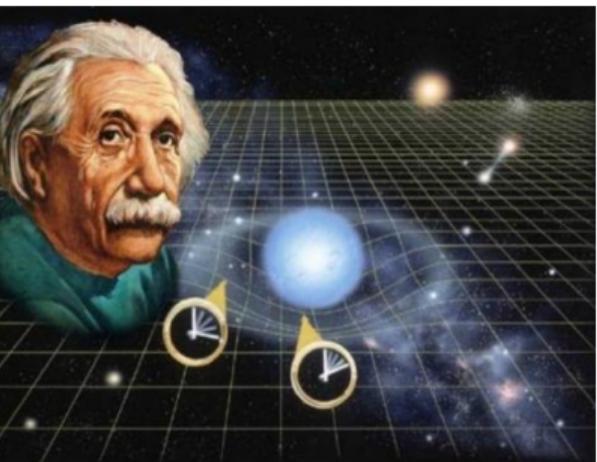
К примеру я бы мог ответить что это способ
упорядочить события
Или же, более образованные люди могут вспомнить
теорию
пространства-времени Ньютона

Time is ...



MYSTERY SOLVED

Way of arrange the order of events



К примеру я бы мог ответить что это способ
упорядочить события
Или же, более образованные люди могут вспомнить
теорию
пространства-времени Ньютона

Definition of time

Compact and robust definition of time has proved to be remarkably tricky and elusive.

- What clocks measure (attr. to physicists Albert Einstein, Donald Ivey, and others)
- What prevents everything from happening at once (physicist John Wheeler and others)
- A linear continuum of instants (philosopher Adolf Grünbaum)

Но чем дольше я искал, тем больше понимал что
Однако, полное и целостное определение времени
до сих пор нерешенная задача
Многие светлые умы пытались. Физики, Философы,
но до сих пор нет определения которое всем
подходит

Religion



Wikipedia

In Zurvanism, Zurvan was perceived as the god of infinite time and space and was aka ("one", "alone").

В поисках интересных определений времени мне даже удалось наткнуться на реально существующую религию в которой Бесконечное Время представляется богом. Сильно, не правда ли

Time is something we deal with every day, and something that everyone thinks they **understand**

Получается довольно забавно. Мы не знаем точно что такое время, однако работаем с ним каждый день.

И чем больше мы с ним работаем тем нам больше кажется что мы его понимаем.

Но это не так



Game time!

И чтобы в этом еще раз убедиться, давайте попробуем пройти небольшую викторину
Викторину о времени
Сейчас будет несколько вопросов и если вам кажется что вы знаете ответ...

Round 1

There are always
24 hours in a day

Верите ли вы
Правда ли
Согласны ли вы с тем

Round 1

There are always
24 hours in a day

Daylight saving time



Верите ли вы
Правда ли
Согласны ли вы с тем

Minute could be longer than
60 seconds

Поскольку вращение Земли постепенно замедляется, разница между средними солнечными сутками и сутками в системе СИ (составляющими ровно 24 часа) в среднем растёт

Round 2

Minute could be longer than
60 seconds

Leap Second



Поскольку вращение Земли постепенно замедляется, разница между средними солнечными сутками и сутками в системе СИ (составляющими ровно 24 часа) в среднем растёт

A month not always
ends in the same year it started

Нет, тут все зависит от календаря и от самого понятия "год"
Финансовый год, китайский новый год
Обычно финансовый год не совпадает с календарным. Например, в США финансовый год установлен с 1 октября по 30 сентября, в Японии — с 1 апреля по 31 марта

A month not always
ends in the same year it started



- Fiscal year
- Chinese Year

Нет, тут все зависит от календаря и от самого понятия "год"

Финансовый год, китайский новый год

Обычно финансовый год не совпадает с календарным. Например, в США финансовый год установлен с 1 октября по 30 сентября, в Японии — с 1 апреля по 31 марта

Round 4

Month could have
less than 20 days

Верите ли вы
Правда ли
Согласны ли вы с тем

Round 4

Month could have
less than 20 days

September 1752 had
19 days in British
Empire



Верите ли вы
Правда ли
Согласны ли вы с тем

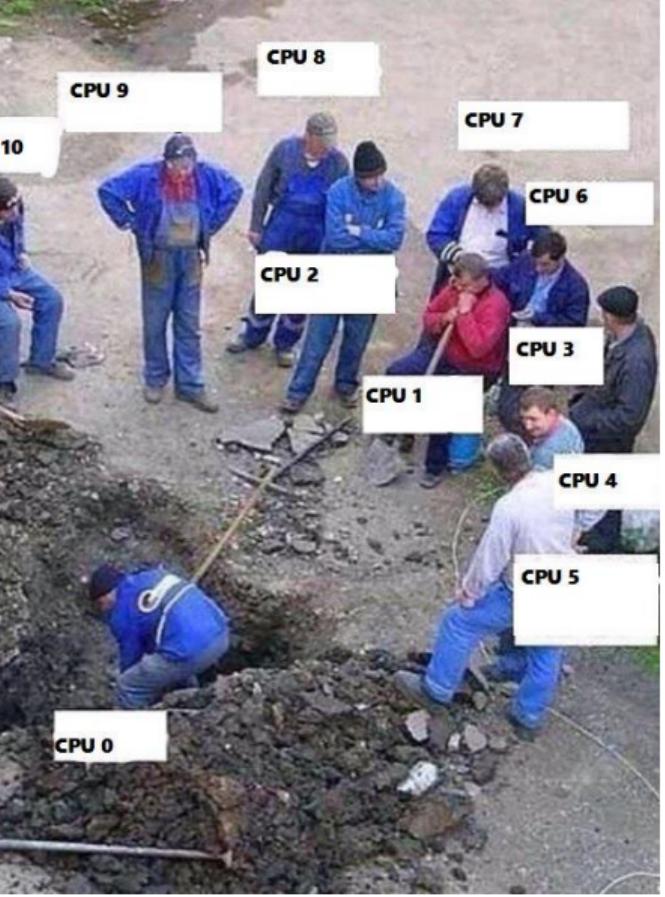
Everyone's do it wrong!?



Вот так вот не просто. И под час меня одолевает мысль что мало кто уделяет внимание деталям работы со временем.
И в жизни у меня случались ситуации которые это доказывают. Сейчас расскажу на примере



У нас завис кластер. Много машин, разом
Мы такие: воу, воу, это точно не мы накосячили
И удивительно что в тот раз это были не мы Мы
начали проверять логи, мониторинг..



У нас завис кластер. Много машин, разом
Мы такие: воу, воу, это точно не мы накосячили
И удивительно что в тот раз это были не мы Мы
начали проверять логи, мониторинг..

bugzilla.redhat.com/show_bug.cgi

Red Hat Bugzilla – Bug 479765

New Q ▾ My Links ▾ Help ▾

Bug 479765 - Leap second message can hang the kernel

Status: CLOSED ERRATA
Alias: None
Product: Red Hat Enterprise Linux 5
Component: kernel 
Sub Component: (Show other bugs)
Version: 5.2
Hardware: All Linux
Priority: high
Severity: medium
Target Milestone: rc
Target Release: ---
Assignee: Prarit Bhargava
QA Contact: Red Hat Kernel QE team
Docs Contact:
URL:
Whiteboard:
Keywords: Reopened, ZStream
Duplicates (1): [800280](#) (view as bug list)
Depends On:
Blocks: [1300182](#) [483701](#) [486920](#) [801794](#)
TreeView+ depends on / blocked

Java Code

```
public class ThreadSleep {  
  
    public static void main(String[] args) throws InterruptedException {  
        long start = System.currentTimeMillis();  
        Thread.sleep(2000);  
        System.out.println("Sleep time in ms = "+(System.currentTimeMillis()-  
start));  
  
    }  
  
}
```

ORACLE® Java Bug Database

Search

Oracle Technology Network > Java > Java SE > Community > Bug Database

JDK-6900441 : PlatformEvent.park(millis) on Linux could still be affected by changes to the time-of-day clock

Type: Bug Priority: P3 Submitted: 2009-11-11
Component: hotspot Status: Closed Updated: 2015-11-27
Sub-Component: runtime Resolution: Fixed Resolved: 2013-09-24
Affected Version: e5.0u21,hs24,hs25,6,6u29,7 OS: linux,linux_ubuntu
CPU: generic,x86,ppc

Versions (Unresolved/Resolved/Fixed)

| JDK 6 | JDK 7 | JDK 8 | Other |
|--|--|--|--|
| 6u71 Fixed | 7u60 Fixed | 8 b109 Fixed | hs25 Fixed |

Related Reports

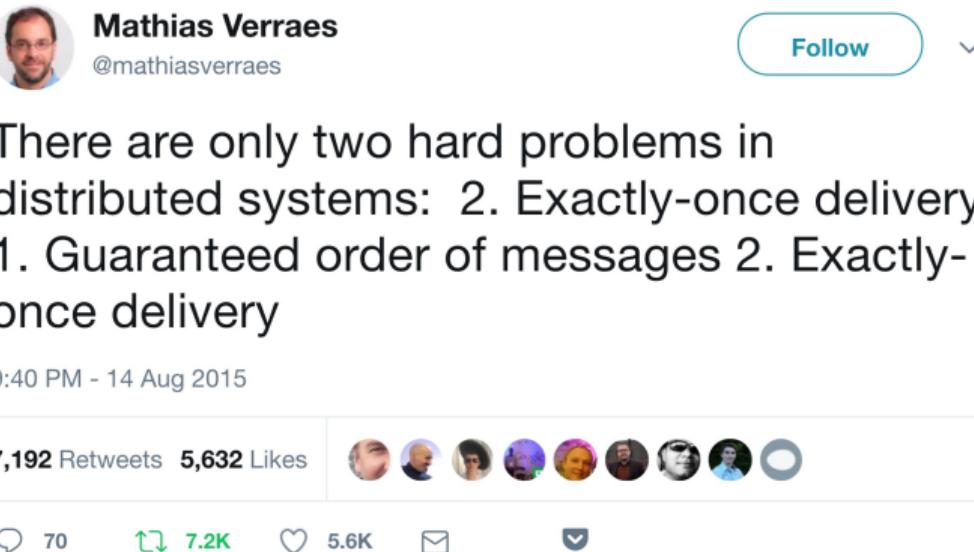
Duplicate : JDK-8024036 - Thread.sleep(long) is not immune to system time shifts with Linux kernel 3.7.10
Duplicate : JDK-7139684 - ScheduledExecutorService doesn't schedules correctly if sys time drifts back
Relates : JDK-8029453 - java/util/concurrent/locks/ReentrantLock/TimeoutLockLoops.java failed by timeout
Relates : JDK-8024036 - Thread.sleep(long) is not immune to system time shifts with Linux kernel 3.7.10
Relates : JDK-8144167 - [OS_X] ConditionObject#awaitNanos waits too long if system clock is turned back
Relates : JDK-6546236 - Thread interrupt() of Thread.sleep() can be lost on Solaris due to race with signal handler
Relates : JDK-6311057 - Java Thread.sleep(timeout) is influenced by changes to System time on Linux

Description

This is a continuation of 6311057 which only partially "fixed" the problem because Linux itself is operating incorrectly in this area. I have it on good authority that "The futex wait implementation is historically wrong." and that "I think with newer kernel/glibc combinations it will be correct."

The real fix here is to change the pthread_cond so that it is associated with the monotonic clock (pthread_condattr_setclock(CLOCK_MONOTONIC) - and with pthread_cond_init being passed the attr object) and to calculate the absolute time as "millis from now" on the monotonic clock. This will make the PlatformEvent::park(millis) code immune to changes in the time-of-day clock.

Distributed environment



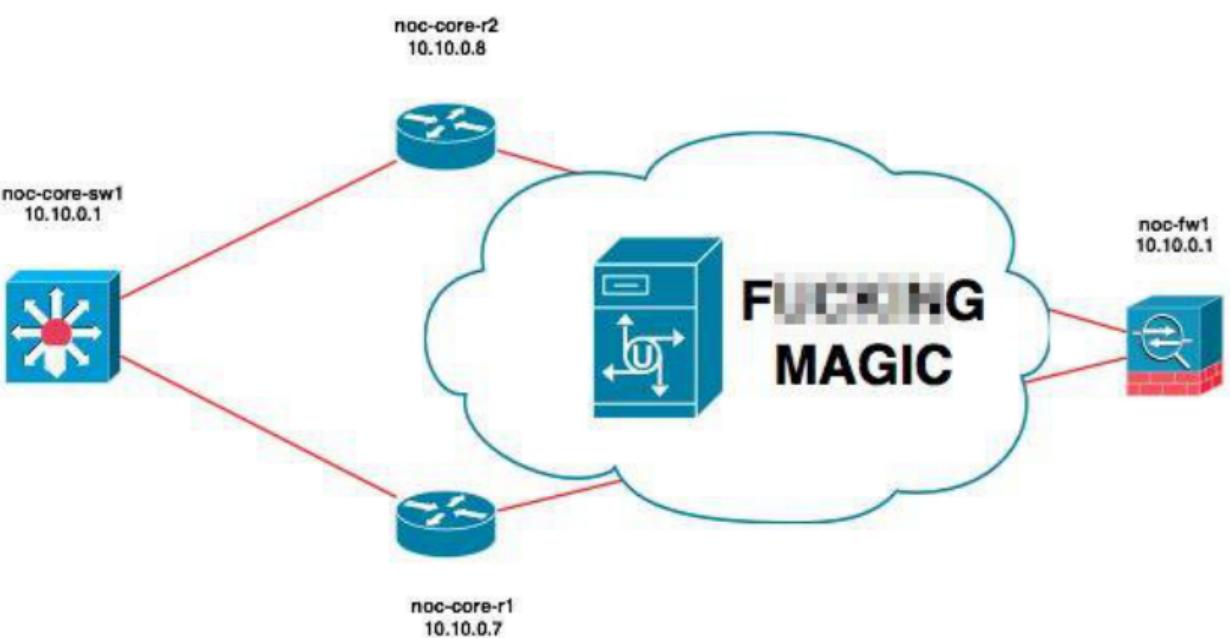
A screenshot of a Twitter post from user @mathiasverraes. The post contains the following text:

There are only two hard problems in distributed systems: 2. Exactly-once delivery
1. Guaranteed order of messages 2. Exactly-once delivery

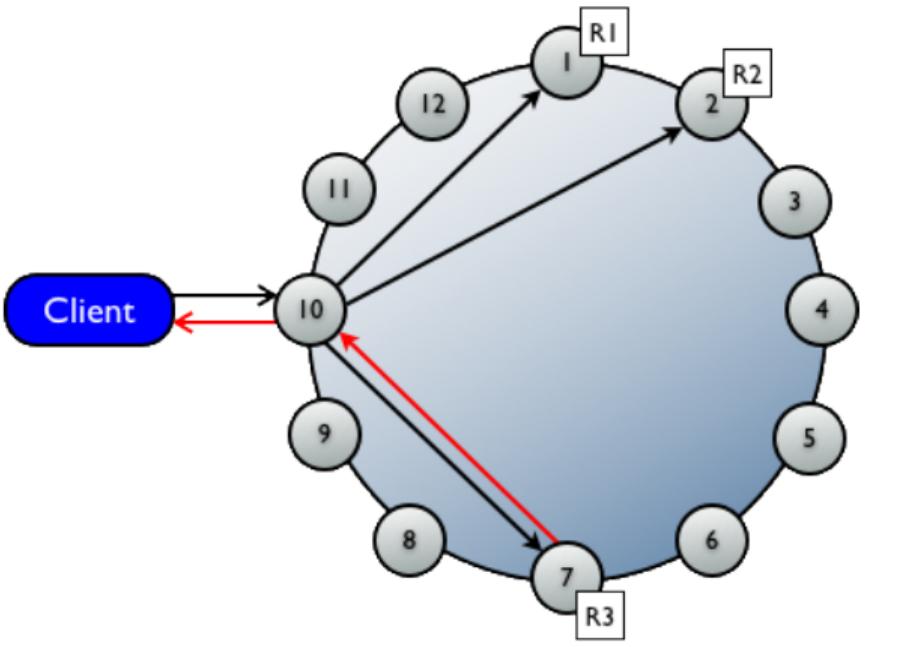
The post was made at 9:40 PM - 14 Aug 2015. It has 7,192 Retweets and 5,632 Likes. Below the tweet are standard Twitter interaction icons for replies, retweets, likes, and shares.

В итоге что имеем, даже серьезные, крупные проекты допускают ошибки в этой области
Но все становится еще хуже когда мы начинаем работать с распределенными системами
А в них у нас есть 2 основные проблемы как обеспечить доставку сообщения один и только один раз а так же как обеспечить порядок обработки сообщений, гарантировать его

Distributed System Topology



И вот если с гарантией доставки время нам помочь не может то с последовательностью событий которые прилетают к нам в распределенную систему кажется что что-то можно придумать... у нас ведь есть часы



- Peer to Peer Database
- Masterless Architecture

Ну вот смотрите, к примеру, у нас есть кассандра
Клиенты могут общаться с каждым компонентом
этой системы
Он подключается и работает с базой. С базой как
единным целым
И клиентов может быть много, но даже в таком случае
база должна сохранить консистентное/целостное
состояние

I am the Law and Order



Needed to establish the order of events that have occurred in the system or when they will occur in the future

- И под консистенцией мы обычно подразумеваем набор правил
- Эти правила обычно позволяют тем или иным способом упорядочить историю событий в распределенной системе
- И в результате, если мы добавим правил

- Maintain consistency
- Build reliable systems
- Debug [Resumption of execution]
- Build Consensus Algorithms

Вот если бы все события приходящие в систему мы бы могли расположить последовательно во времени

- поддерживать целостность данных
- строить предсказуемые системы которые могут предоставлять гарантии по процессу обработки
- для решения задачи консенсуса

Ну что, попробуем!?

Let's try Common Clock [Wall Clock]!

Начнем с самого простого, казалось бы, решения
Будем использовать обычные часы которые есть у
нас на каждой машине. Если мы работаем в рамках
одного сервера

Common Clock

We can reliably define

- simultaneous: all events that happen between clock ticks

Определить отношение

Сказать что одно было раньше другого или одно было после другого

Common Clock

We can reliably define

- simultaneous: all events that happen between clock ticks
- before: an event that happens in a previous clock tick
- after: an event that happens in a subsequent clock tick

Определить отношение

Сказать что одно было раньше другого или одно было после другого

But there's **no Common Clock** in distributed systems!

- No shared memory
- No common clock

Общей разделяемой памяти и нету общих часов в отличии от ситуации когда мы работаем в рамках одного сервера

Стоп Стоп скажете вы. Там же были алгоритмы и протоколы для решения этой проблемы

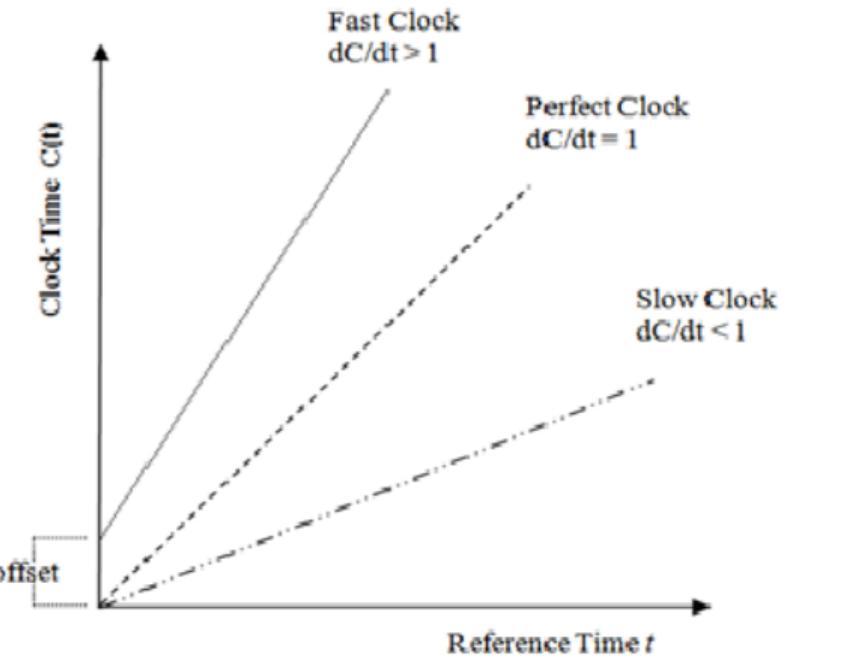
Wall Clock

Let's use 'Wall Clocks' and ... NTP!?



- Да, давайте попробуем. Попробуем обычные часы и их синхронизацию
- А синхронизация у нас борется с 2мя основными проблемами, Drift & Skew
- CHOC и CKOC

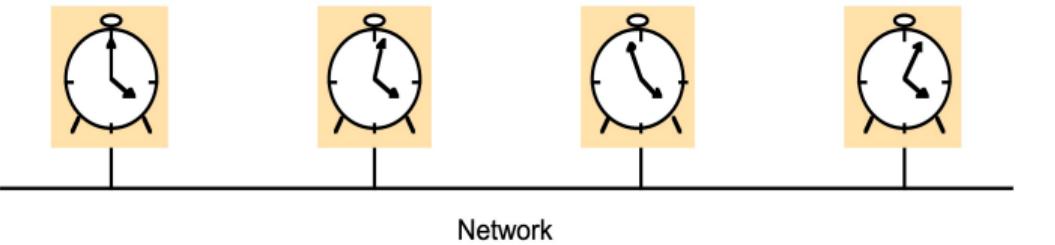
Clock Drift



Ordinary clocks drift by about 1 sec in 10 days

We can't have perfect clocks

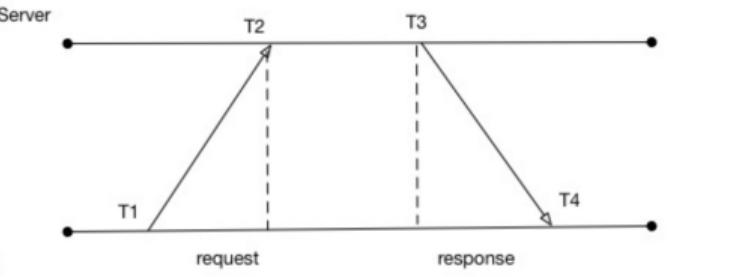
Clock Skew



Skew: the difference between the times on two clocks (at any instant)

К примеру у нас есть кластер серверов. И в нем всегда будут расхождения в показании во времени между серверами. Это снос. С ним тоже нужно бороться

I'm running NTP. I'm good



The challenge with this approach is that there is a delay in the transmission from the time server to the client receiving the time update

This **delay is not constant for all requests**. Some request may be faster and others slower

- И казалось бы для этого у нас есть протоколы. такие как NTP
- Протоколы, призванные синхронизировать время на машинах в сети
- И они не плохо справляются
- Однако, все эти протоколы не идеальны. Они основаны на замерении задержек между сервером времени и машиной на которой нужно настроить время
- Да ну, скажете вы. Нам и такого хватит

I'm running NTP. I'm good

600000 events per second

Допустим они приходят равномерно в течении секунды

I'm running NTP. I'm good

$1 / 600000 = 0.0000016 \dots \text{microsecond?}$

Допустим они приходят равномерно в течении секунды

I'm running NTP. I'm good

| Protocol | Media | Sync Accuracy |
|----------|----------|---------------------|
| NTP | Ethernet | 50-100 milliseconds |
| IRIG-B | Coaxial | 1-10 microseconds |
| PTP | Ethernet | 20-100 nanoseconds |

* Choosing the correct Time Synchronization Protocol and incorporating the 1756-TIME module into your Application

By: Josh Matson

Нам уже не подходит NTP, да и PTP в реальности не всегда сможет помочь
И я думаю вы уже поняли к чему я клоню.
У них у всех есть один общий недостаток

Boom!



Correction of time does not mean that all machines agree on time, it just means they are much **closer to each other on average**

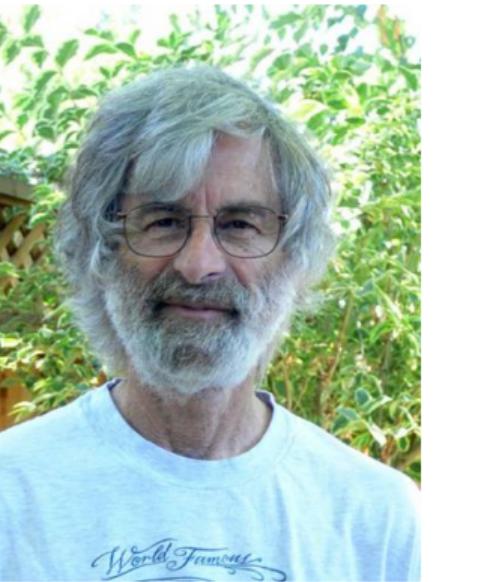
А все потому что алгоритмы не могут синхронизовать часы точно, существует лишь возможность уменьшить расхождение между часами на разных машинах

Wall clocks solutions require

- Sophisticated algorithms or special hardware
- Monitoring

Нам придется использовать сложные алгоритмы, а еще и пристально следить за нашими системами чтобы СКОС не появился и все не развалилось
А что если не использовать настенные часы?

Leslie Lamport



Lamport, L. (1978). "Time, clocks, and the ordering of events in a distributed system"

"Время, часы и порядок событий в распределенной системе"

- The important contribution of Lamport is that in a distributed system, **clocks need not be synchronized absolutely**
- It is **not important** that all **processes agree on what the actual time is, but that they agree on the order in which events occur**
- Happens Before

основные идеи

- не нужно синхронизировать часы идеально.
даже без точной синхронизации
- процессам не обязательно договариваться о
точном времени чтобы договориться об общей
истории событий
- задает отношение happens before. Отношение,
которое лежит в основе так называемых часов
Лэмпорта

Happens Before

The relation ' \rightarrow ' on the set of events of a system is a smallest relation satisfying three conditions

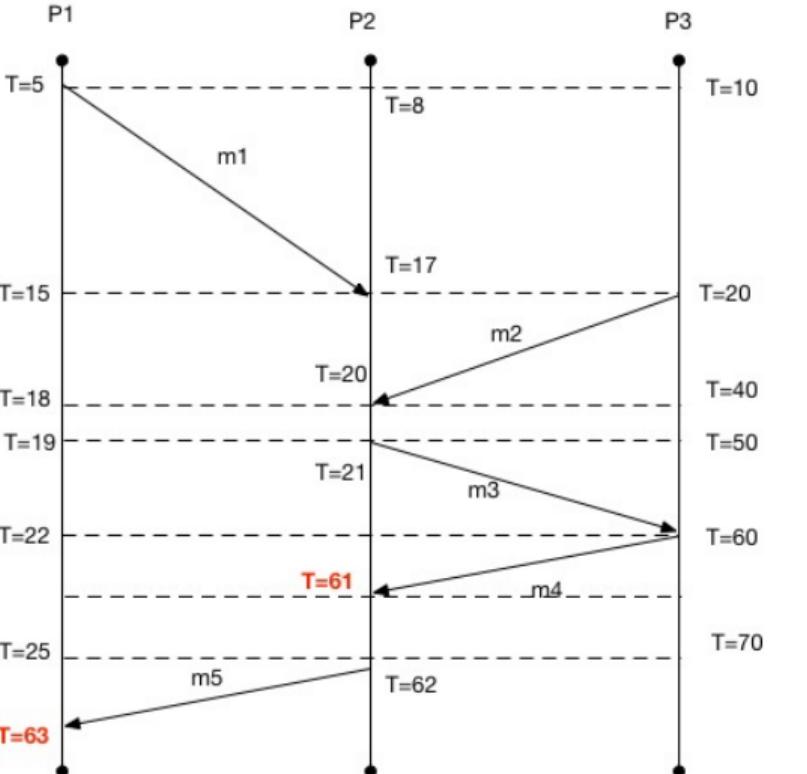
- If a, b are events **in the same process**, and a comes before b , then $a \rightarrow b$
- If a is the sending of a message by one process and b is the receipt of the same message by another process, then $a \rightarrow b$
- Transitive. If $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$. Two distinct events a and b are said to be concurrent if $a \not\rightarrow b$ and $b \not\rightarrow a$

If $a \rightarrow b$ happens between **two process**, and events x and y occur on another set of processes and these two sets of processes don't exchange messages then: we cannot say whether $x \rightarrow y$ or $y \rightarrow x$ from the perspective of the first set of processes

если событие а это "посылка сообщения" а любому процессу, а б это событие когда это сообщение было получено
outcome
Этого достаточно чтобы создать логические часы.
Часы Лэмпорта

Lamport Clocks

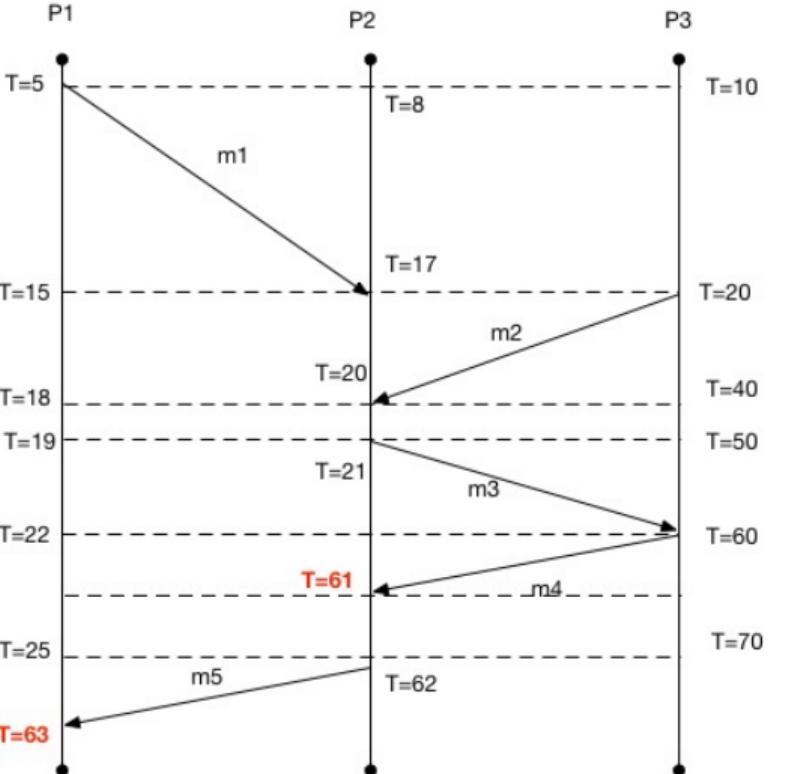
Example



- When a message is transmitted from $P_1 \rightarrow P_2$, P_1 will encode the send time into the message

транзакции

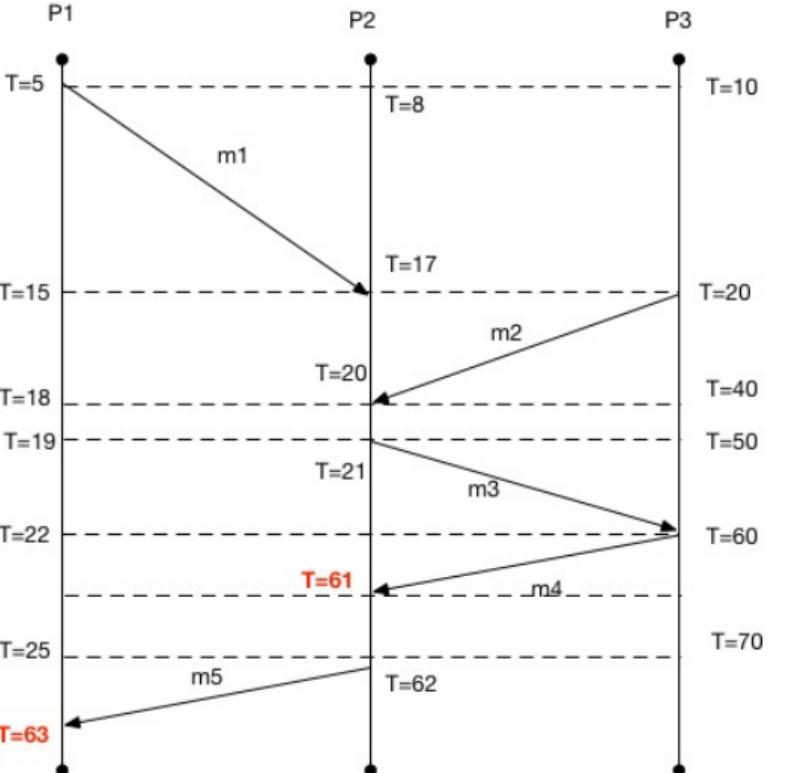
Example



- When a message is transmitted from $P_1 \rightarrow P_2$, P_1 will encode the send time into the message
- When P_2 receives the message, it will record the time of receipt

транзакции

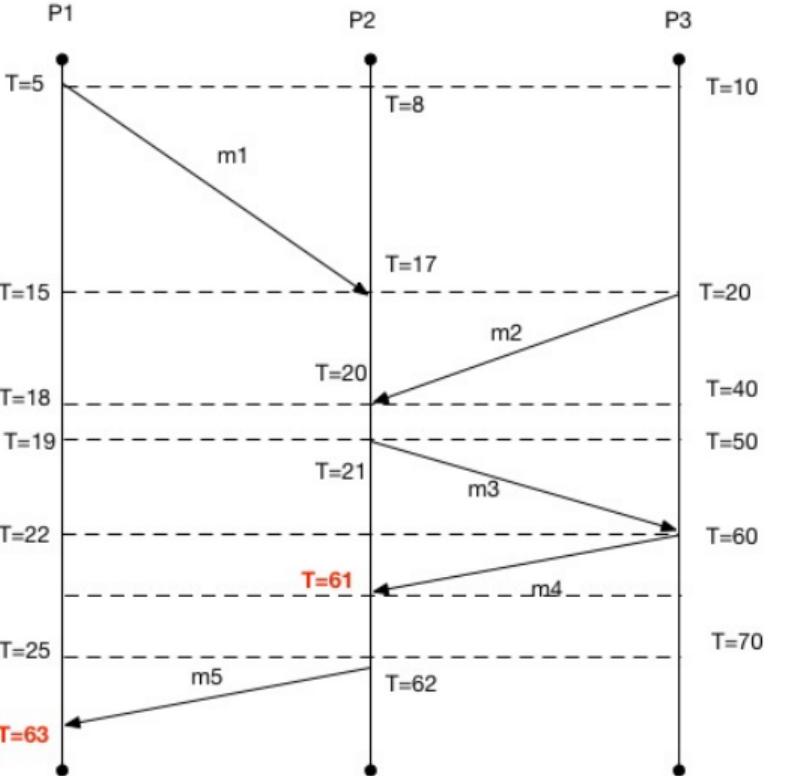
Example



- When a message is transmitted from $P_1 \rightarrow P_2$, P_1 will encode the send time into the message
- When P_2 receives the message, it will record the time of receipt
- If the time at P_2 is already greater than the send time, then no action is required for P_2

транзакции

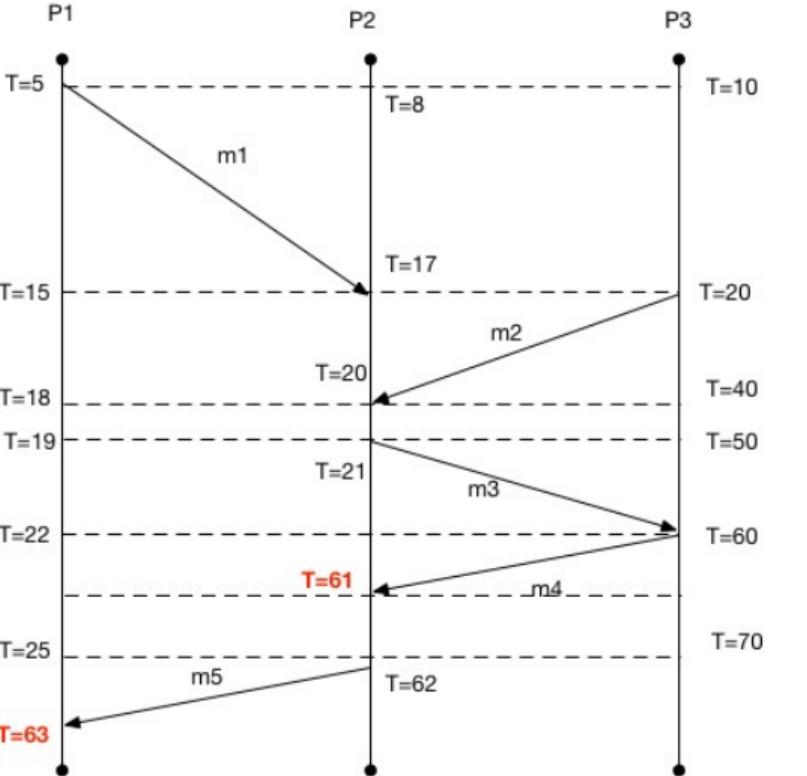
Example



- When a message is transmitted from $P_1 \rightarrow P_2$, P_1 will encode the send time into the message
- When P_2 receives the message, it will record the time of receipt
- If the time at P_2 is already greater than the send time, then no action is required for P_2
- If P_2 discovers that the time of receipt is before the send time, P_2 will update its software clock to be one greater than the send time

транзакции

Example



- When a message is transmitted from $P1 \rightarrow P2$, $P1$ will encode the send time into the message
- When $P2$ receives the message, it will record the time of receipt
- If the time at $P2$ is already greater than the send time, then no action is required for $P2$
- If $P2$ discovers that the time of receipt is before the send time, $P2$ will update its software clock to be one greater than the send time
- “happens-before” relationship is preserved with this actions

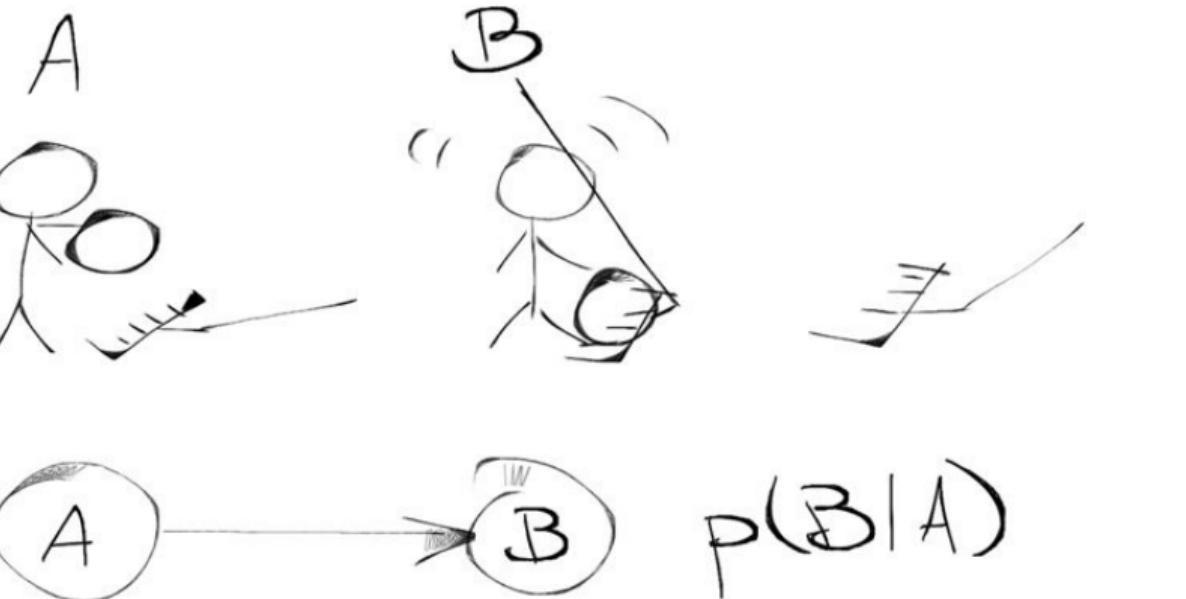
транзакции

Lamport Clocks Limitations

With Lamport's clocks nothing can be said about
the actual time of a and b.

При использовании часов Лемпорта мы ничего не можем сказать о реальном времени, когда происходили события
Эти часы оторваны от реального времени

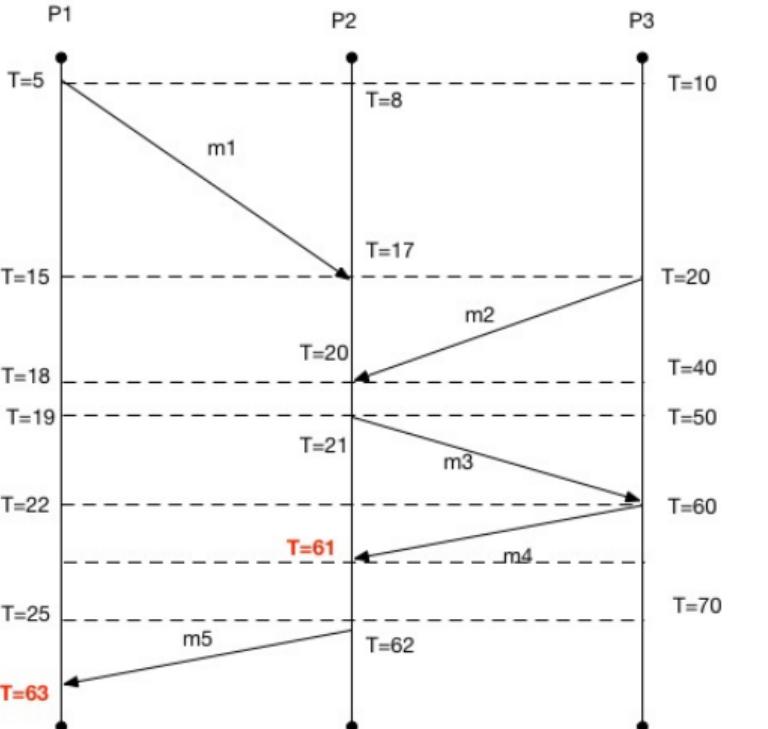
Causality



Causal relations
An example of a causal relation and its representation in a graph.

Credits: C. Giarmatzi

Who did when?



- $m1 \rightarrow m3$
- $m2 \rightarrow m3$

Knowing $m1 \rightarrow m3$ and $m2 \rightarrow m3$ shows $m3$ did not cause $m1$ or $m2$ but we cannot say which initiated $m3$

Vector Clocks

by Colin Fidge and Friedemann Mattern in 1988

Усложненные часы Лэмпорта, которые должны решать их проблемы
Посмотрим внимательнее

Vector Clocks

- A vector clock of a system of N processes is **an array/vector** of N logical clocks, **one clock per process**
- A vector clock $VC(a)$ is assigned to an event a
- If $VC(a) < VC(b)$ for events a and b , then event a is known to causally precede b

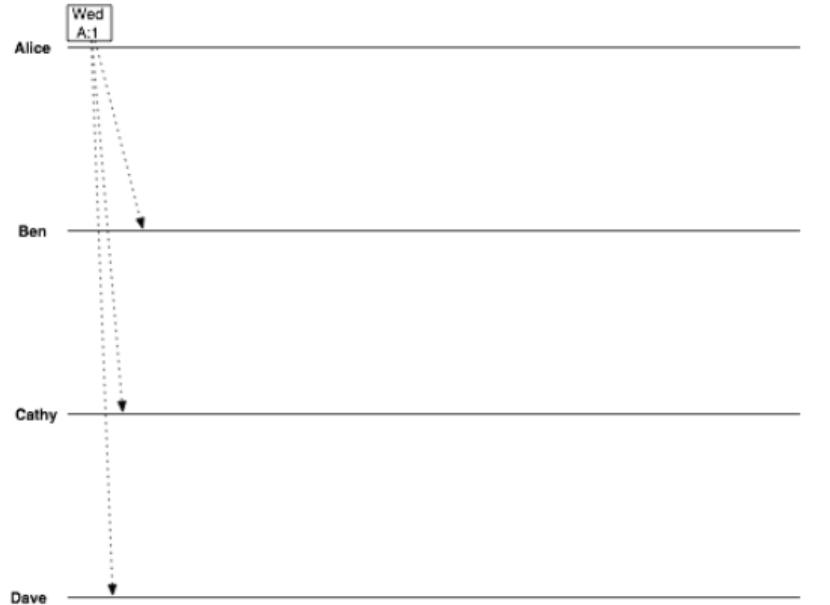
- Векторные часы системы из N процессов состоят из массива размером N логических часов
 - каждому событию присваивается такой массив часов
 - если все логические часы одного вектора/массива меньше

Robbery



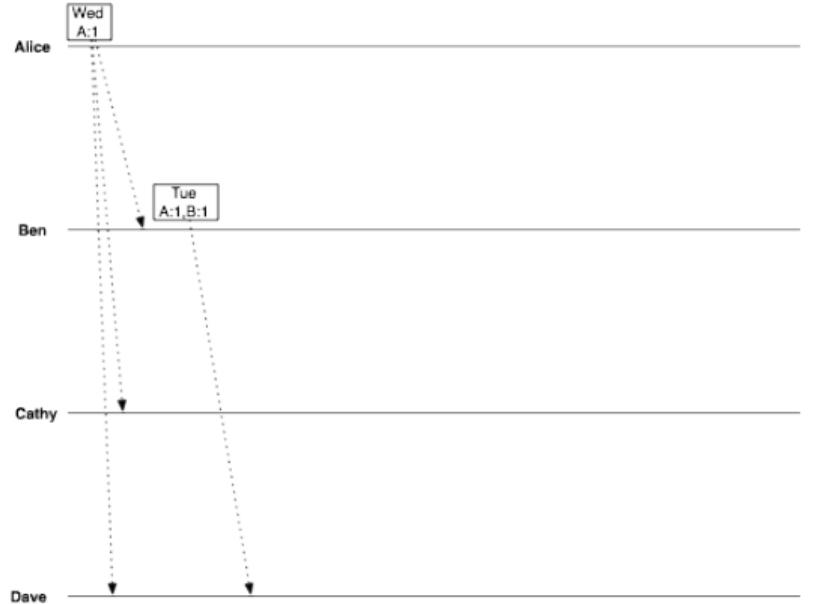
Alice, Ben, Cathy, and Dave want to
rob a bank

- They could send P2P messages
- No leader



```
date = Wednesday  
vclock = Alice:1
```

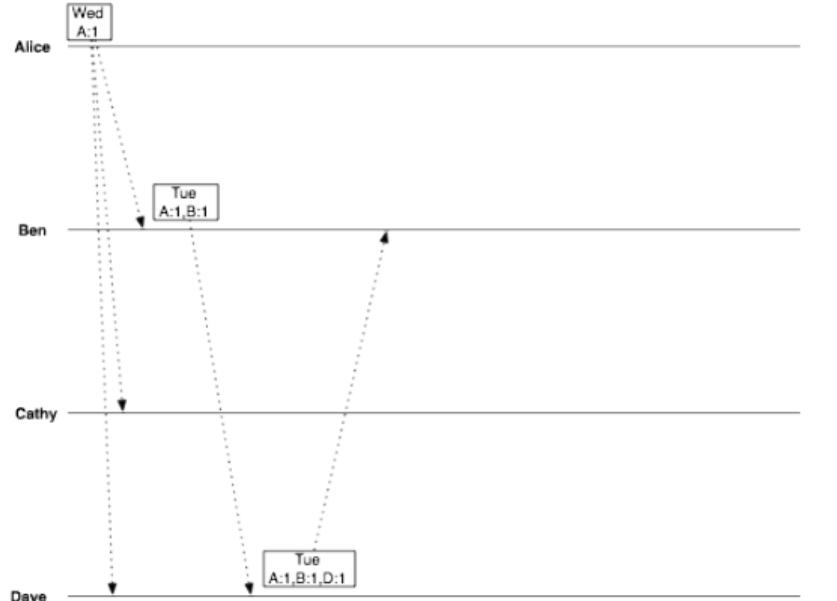
Среда



Ben suggests Tuesday

```
date = Tuesday  
vclock = Alice:1, Ben:1
```

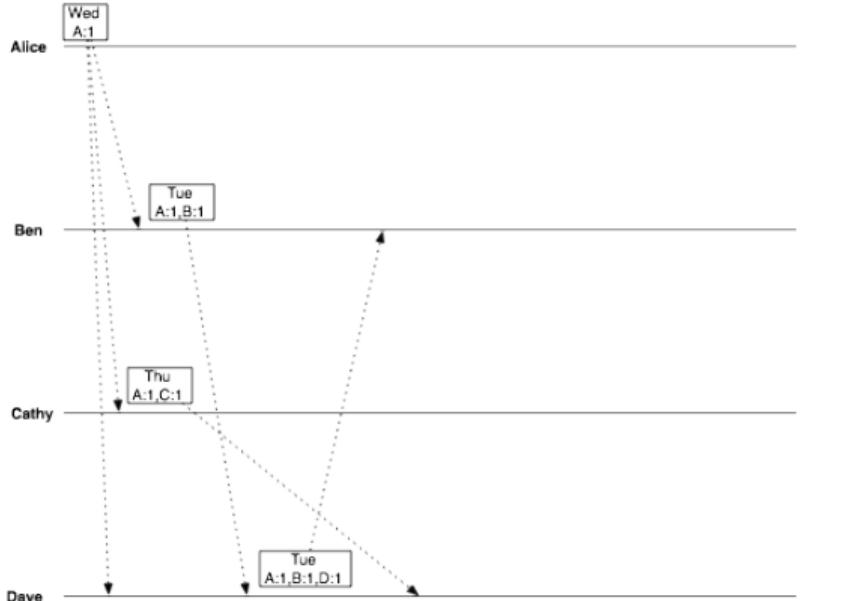
Вторник



Dave replies, confirming Tuesday

```
date = Tuesday  
vclock = Alice:1, Ben:1, Dave:1
```

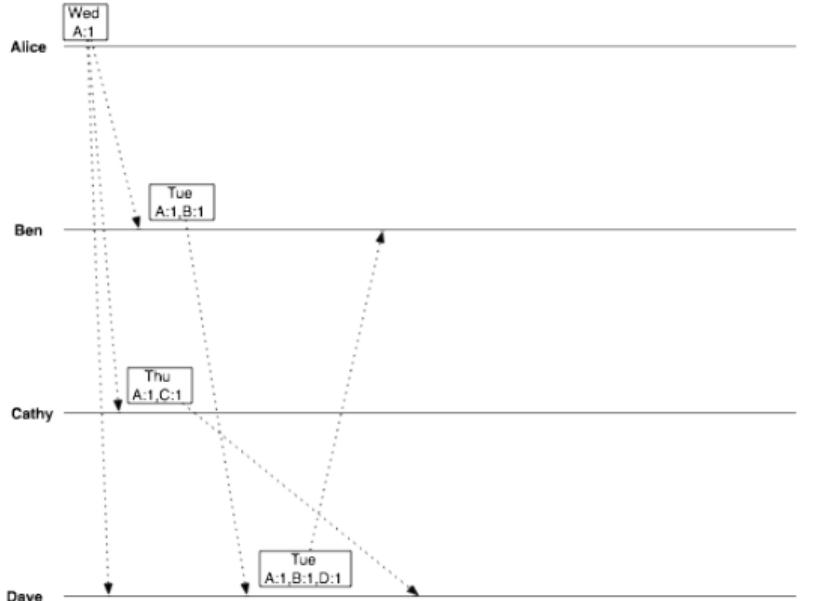
Вторник



Then Cathy joins, suggesting Thursday

```
date = Thursday  
vclock = Alice:1, Cathy:1
```

Четверг



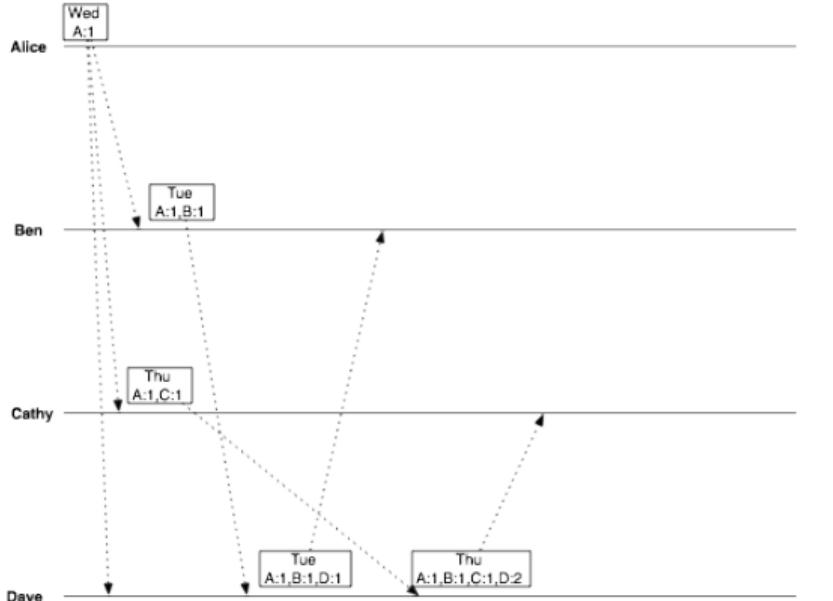
Dave has two conflicting objects

```
date = Tuesday  
vclock = Alice:1, Ben:1, Dave:1
```

and

```
date = Thursday  
vclock = Alice:1, Cathy:1
```

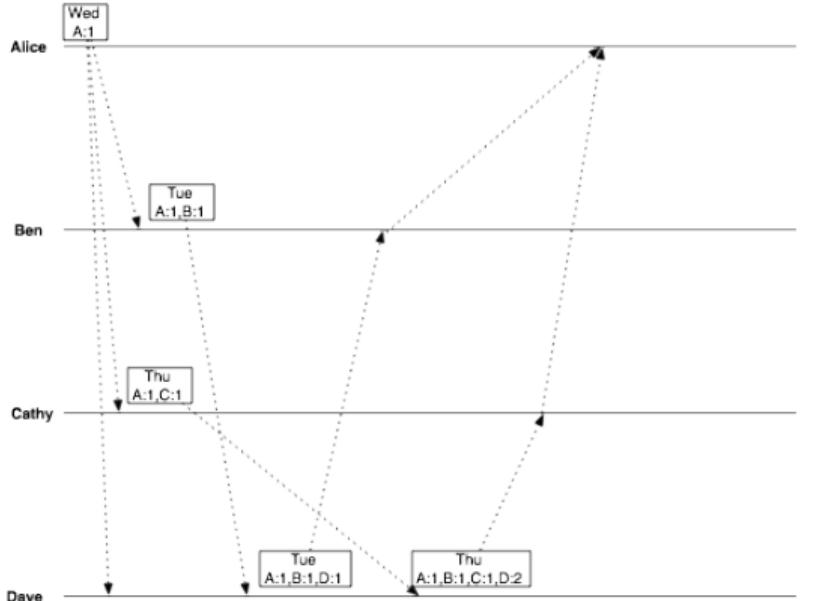
Вторник и четверг



Dave's a reasonable guy, and chooses Thursday.
So he sends this value back to Cathy

```
date = Thursday  
vclock = Alice:1, Ben:1, Cathy:1,  
Dave:2
```

Четверг



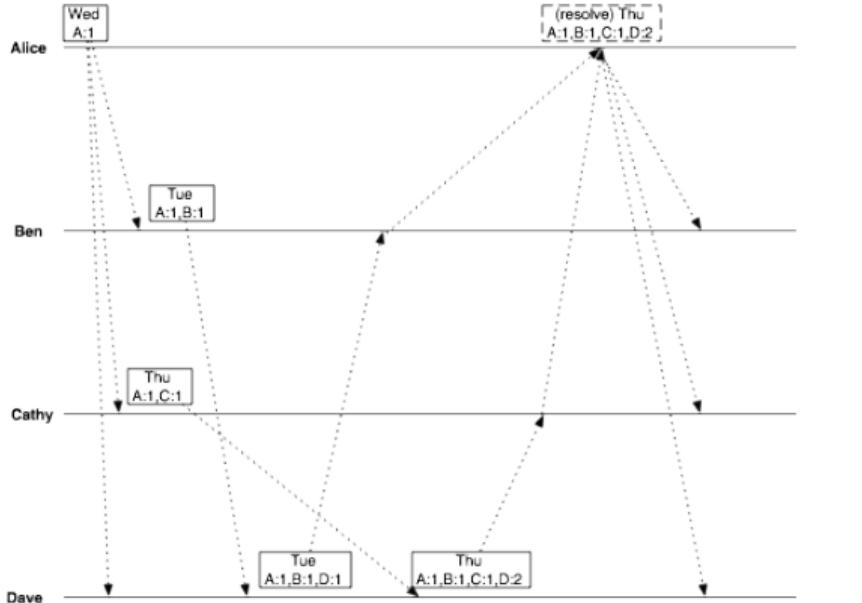
When Alice asks Ben and Cathy for the latest decision, the replies she receives are, from Ben

```
date = Tuesday  
vclock = Alice:1, Ben:1, Dave:1
```

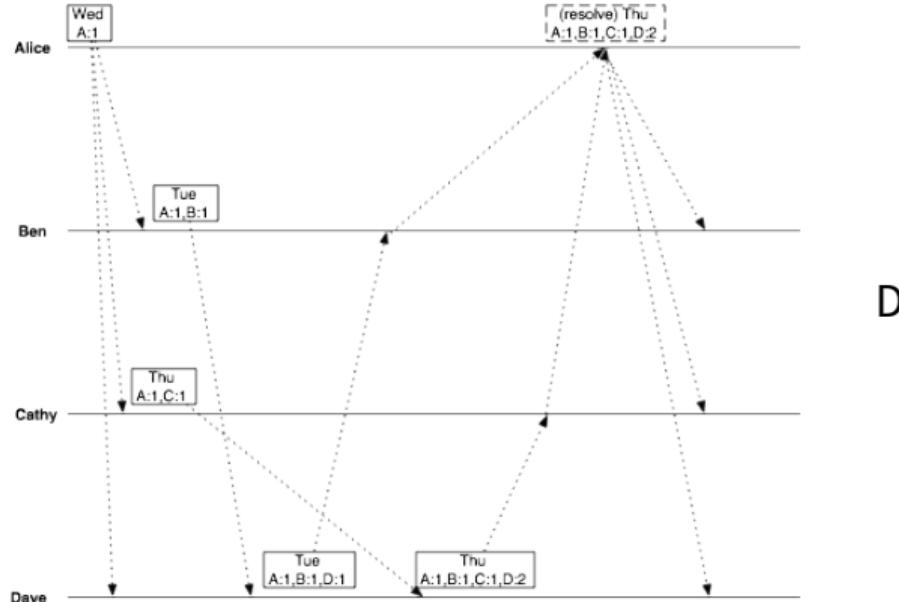
From Cathy and Dave

```
date = Thursday  
vclock = Alice:1, Ben:1, Cathy:1,  
Dave:2
```

Вторник и четверг



All Alice has to do is show Ben the vector clock from Cathy's message, and Ben will know that he has been overruled.



Done. Everybody's in sync



So... What Did We Get Out Of All of This?

Мы посмотрели несколько вариантов решения проблемы определения порядка событий в распределённой системе

Review

- Physical Clocks - hard to keep synchronized
- Logical clocks - can provide some notion of relative events occurrence
- Lamport's logical time
 - happened before defines causal relation
 - these clocks don't capture causality
 - total ordering relation
- Vector Clocks
 - captures causality
 - have a component for each process in the system
 - slow and grow

И все эти решения вы сможете найти под капотом современных распределенных систем
Некоторые используют настенные часы, а другие доверяют только логическим
Дело в том



Think ahead! Think advance!

- Нет плохих решений, нет хороших. Есть решения подходящие и не подходящие
- Если вы строите распределенные системы вы тоже столкнетесь с проблемами синхронизации
- Даже если это сайт за лоад балансером
- И если вы не уделите ей внимание, то придется потом лепить кости и в худшем случае

И вот здесь вот я хотел бы взять паузу
Паузу, ведь мы проделали довольно большой путь,
много чего разобрали
Однако, там осталось еще очень много чего
интересного. На еще одну презентацию, к примеру

So many things to tell you

- GPS Clocks
- Atomic Clocks in Google Spanner Database
- UTC, TAI, TCG and TCB coordinate time standards

А так же остались не затронутыми такие
увлекательные вопросы как
Ну а текущая моя презентация подходит к концу

So many things to tell you

- GPS Clocks
- Atomic Clocks in Google Spanner Database
- UTC, TAI, TCG and TCB coordinate time standards
- Why do clocks run clockwise?
- Why there was no 30 December 2011 in Samoa?

А так же остались не затронутыми такие
увлекательные вопросы как
Ну а текущая моя презентация подходит к концу



Thank you!