



# Time In Distributed Systems

## Pt.1



# Hello



Denis Golovachev



SPB



Saint-Petersburg

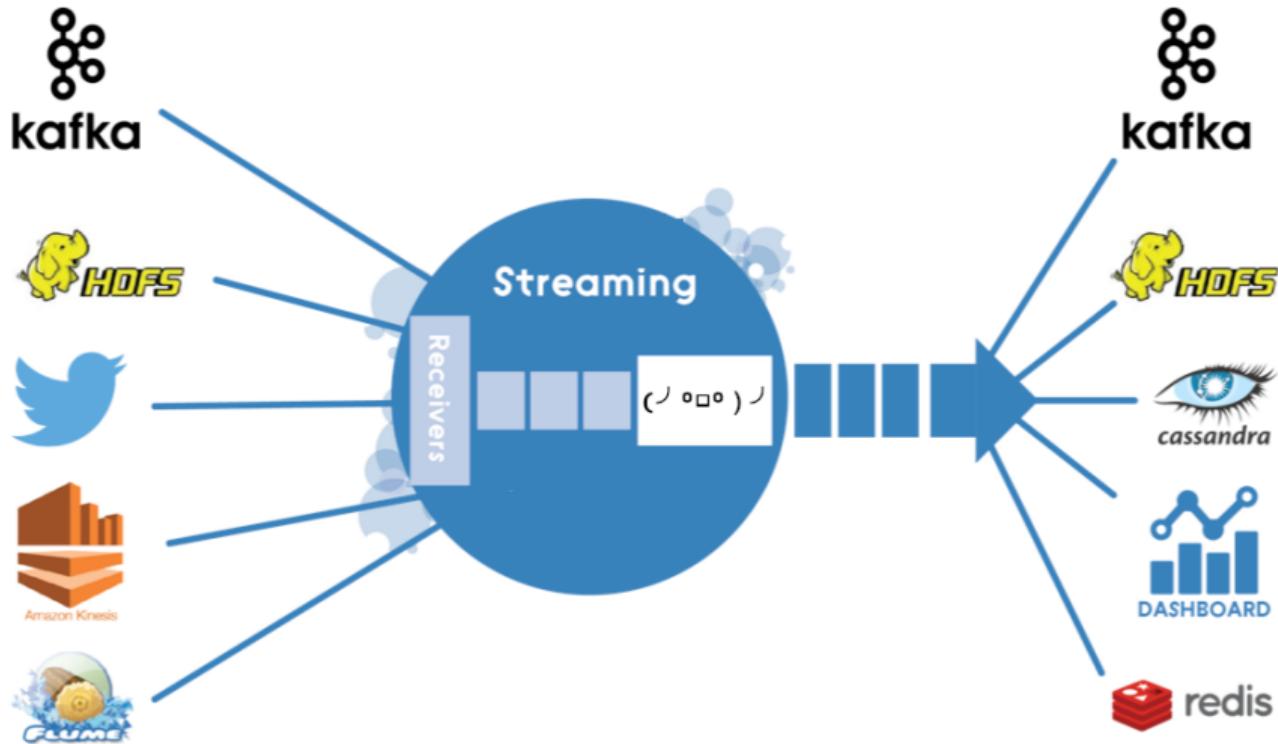




# Software Architect

# Data Streaming









# Big Data School



Grid Dynamics

**Big Data School**



## Professor

# Boring







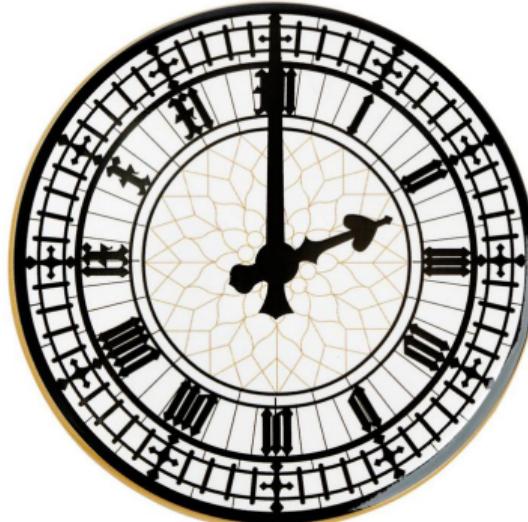
# Time



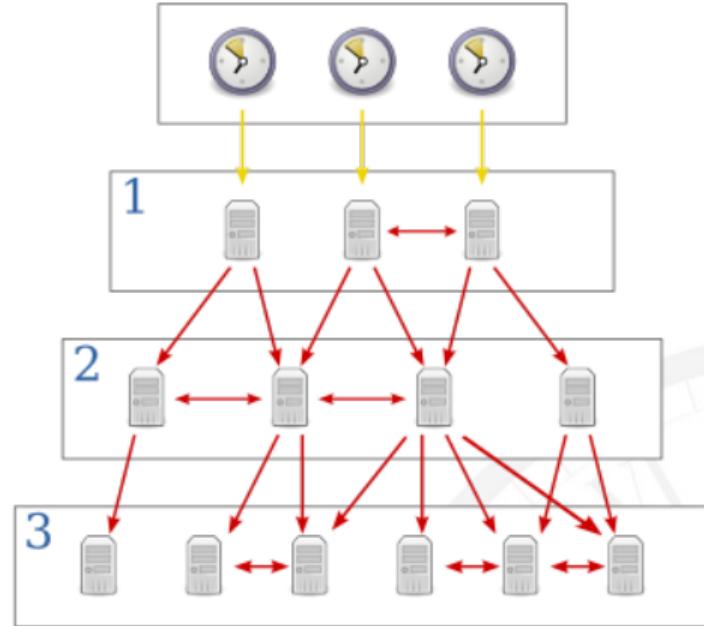
# Time



# Time



# Time



Question

What is time?

# Time is ...



1998

2004

3XXX

## MYSTERY SOLVED

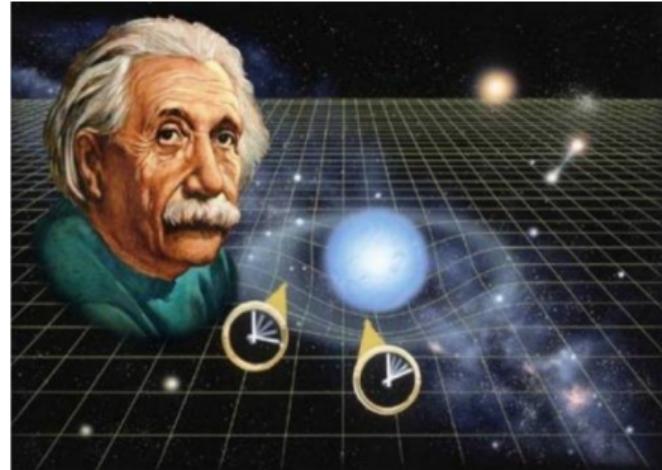
Way of arrange the order of events

# Time is ...



## MYSTERY SOLVED

Way of arrange the order of events



Another dimension

## 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)

Time is ...

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



# Game time!

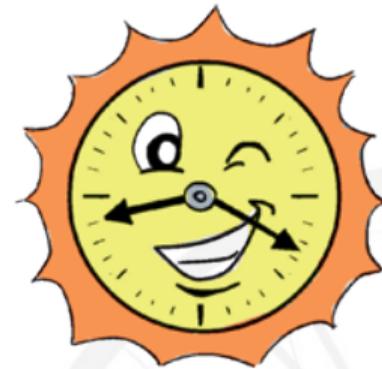
## Round 1

There are always 24 hours in a day

## Round 1

There are always 24 hours in a day

Daylight saving time



*Daylight Saving  
Time Begins*

## Round 2

There are 60 seconds in a minute

## Round 2

There are 60 seconds in a minute

Leap Second



## Round 3

A month always ends in the  
same year it started

## Round 3

A month always ends in the same year it started

- Fiscal year/calendar
- Chinese Year



## Round 4

Month could have less than 20 days

## Round 4

Month could have less than 20 days

September 1752 had  
19 days in British  
Empire

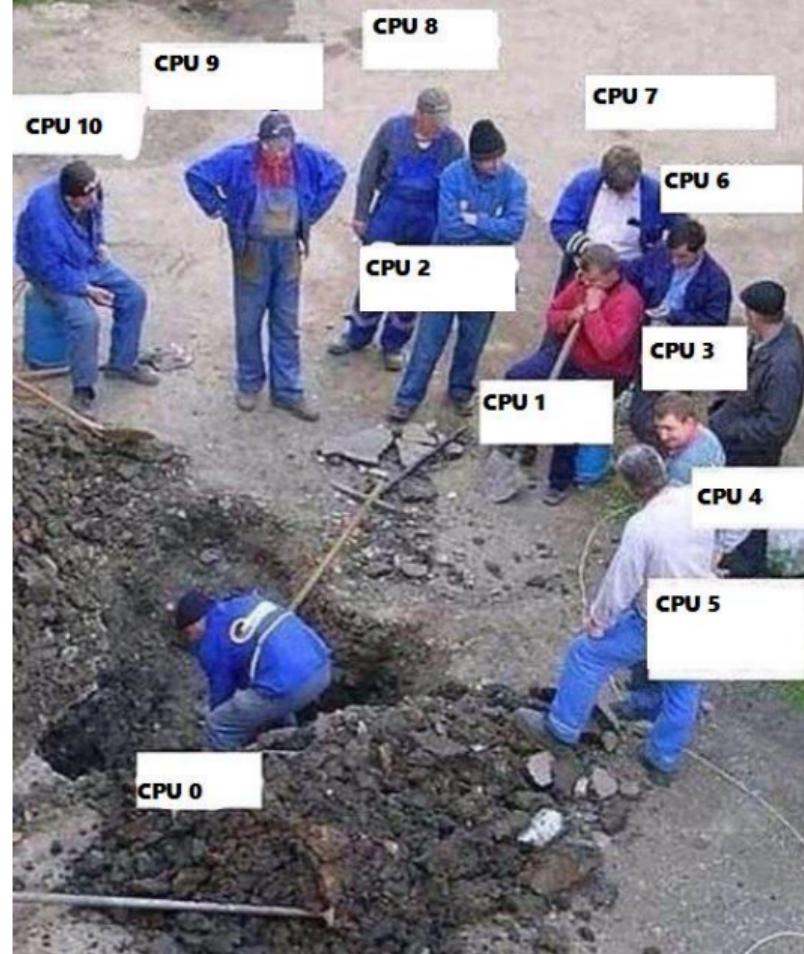


Everyone's do it wrong!



**SERVER DOWN?**





## Red Hat Bugzilla – Bug 479765

[Home](#) [New](#) [Search](#) [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:**

Type a sub-component name

[\(Show other bugs\)](#)**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):** [800289](#) ([view as bug list](#))**Depends On:****Blocks:** 1300182 483701 485920 801794**TreeView+ depends on / blocked****Reported:** 2009-01-12 22:31 UTC by Chris Adams**Modified:** 2018-11-28 20:21 UTC ([History](#))**CC List:** 15 users ([show](#))**Fixed In Version:****Doc Type:** Bug Fix**Doc Text:****Clone Of:****Environment:****Last Closed:** 2009-09-02 08:33:56 UTC**Dependent Products:**

# 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 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 <span>Fixed</span>	7u60 <span>Fixed</span>	8 b109 <span>Fixed</span>	hs25 <span>Fixed</span>

### 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

# Distributed environment



**Mathias Verraes**

@mathiasverraes

Follow



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

9:40 PM - 14 Aug 2015

7,192 Retweets 5,632 Likes



70

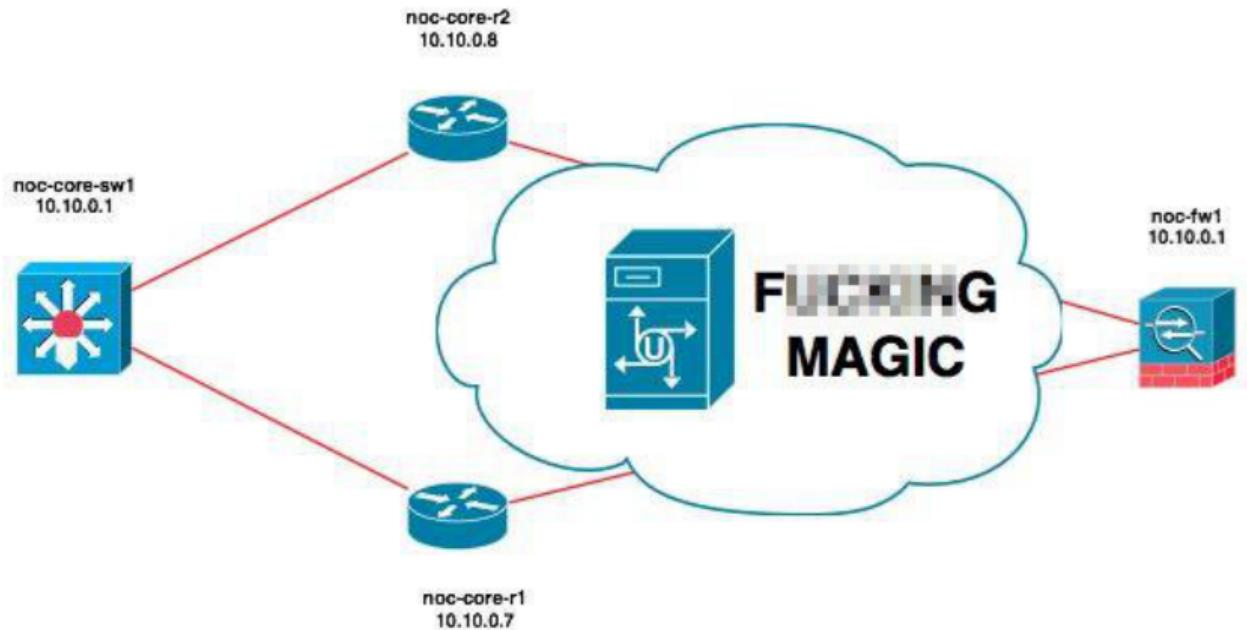


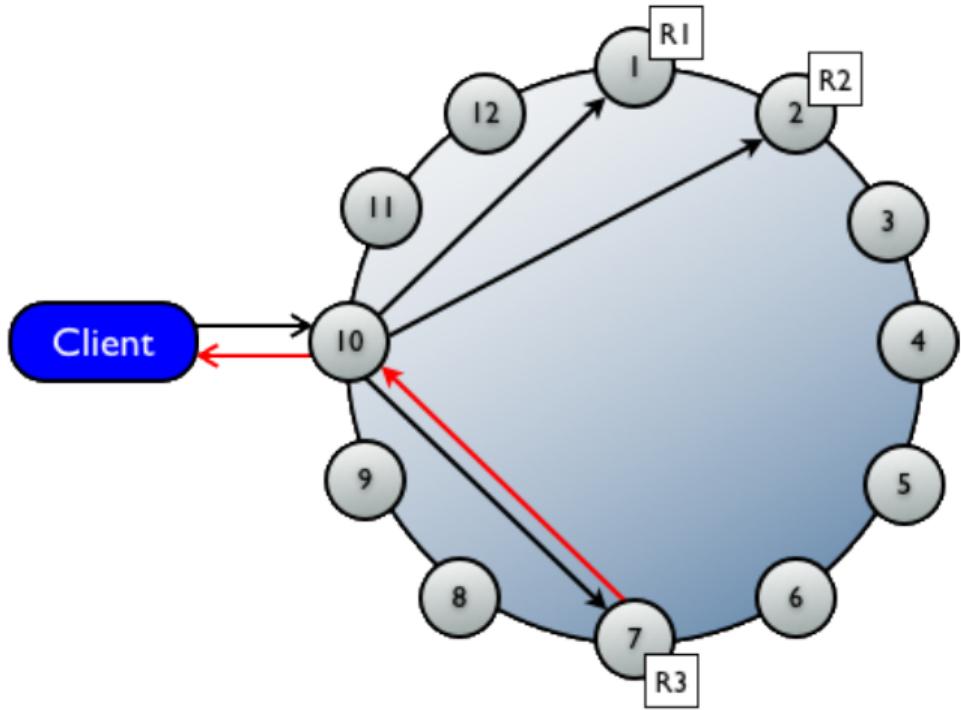
7.2K



5.6K







- Peer to Peer Database
- Masterless Architecture

its difficult to reason about events in this system

# 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

# Order in Chaos

- Maintain consistency
- Build reliable systems
- Mutual exclusions
- Debug [Resumption of execution]

## Common Clock

We can reliably define

- simultaneous: all events that happen between clock ticks

## 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

Synchronization in distributed systems is **hard**

- **No shared memory**
- **No common clock**

# Wall Clock

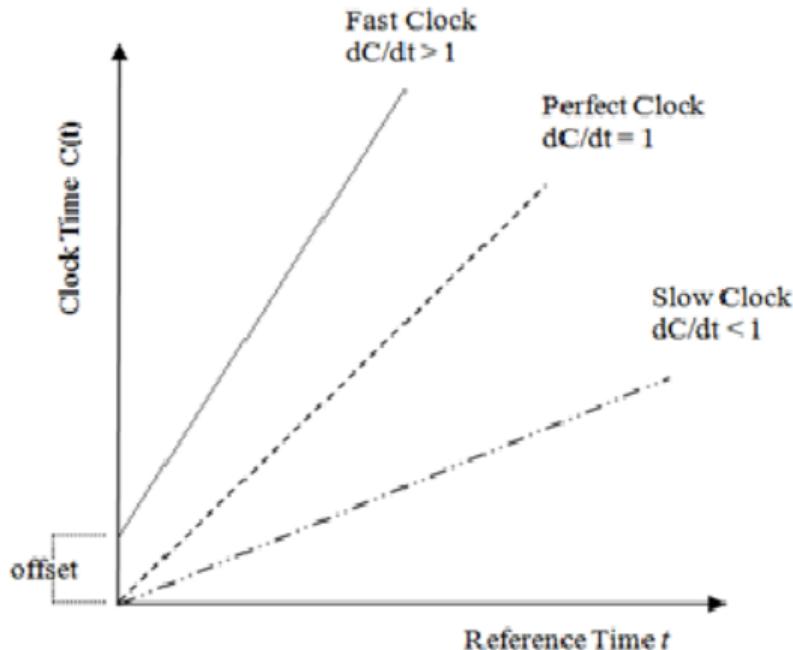
Let's use 'Wall Clocks' and NTP!?



When kept under tension the quartz crystal oscillates at a well-defined frequency

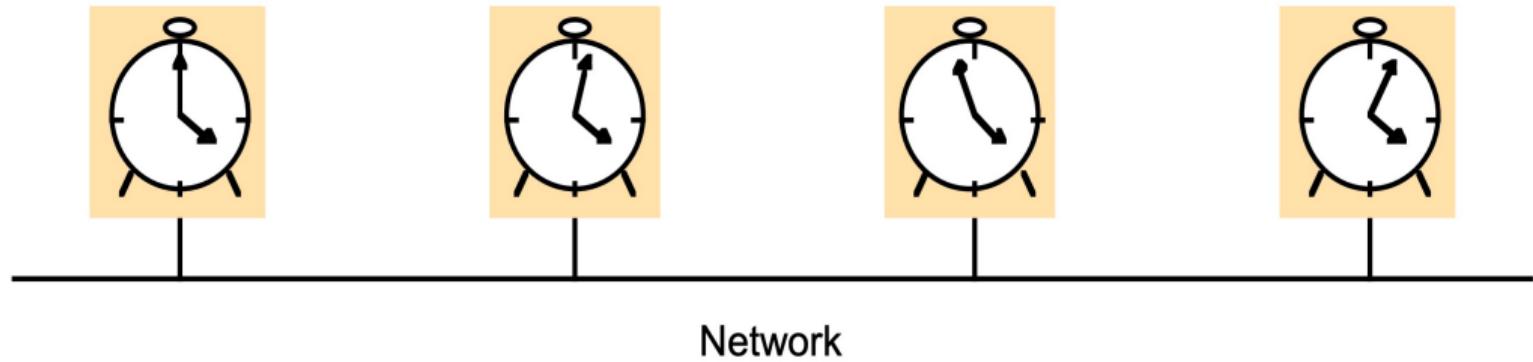
# Clock Drift

We can't have perfect clocks



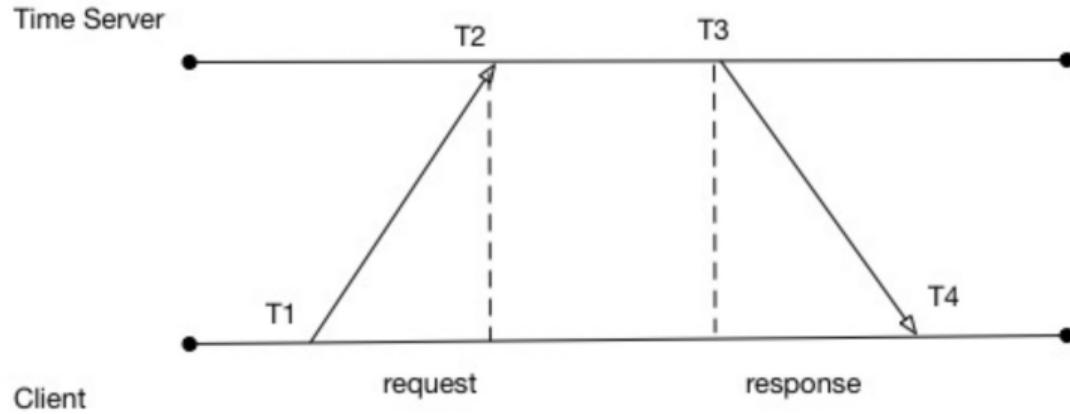
Ordinary clocks drift by about 1 sec in 10 days

# Clock Skew



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

# 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 requests may be faster and others slower.

I'm running NTP. I'm good

600000 events per second

I'm running NTP. I'm good

$1 / 600000 = 0.0000016 \dots$  microsecond?

# 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

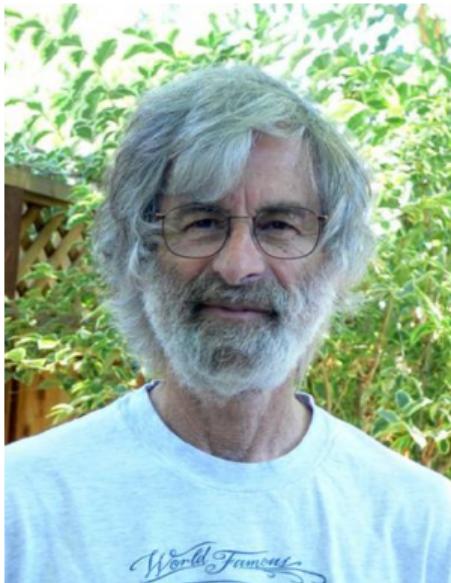
\* Choosing the correct Time Synchronization Protocol and incorporating the 1756-TIME module into your Application  
By: Josh Matson

# 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**

# Leslie Lamport



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

- 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

# Happens Before

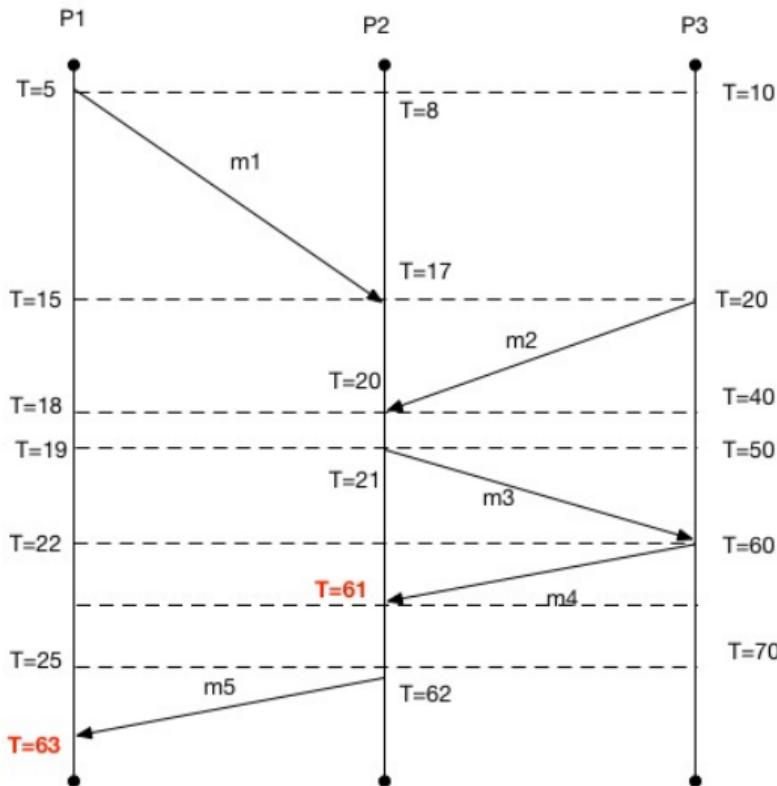
The relation ' $\rightarrow$ ' on the set of events of a system is a smallest relation satisfying three conditions

- If  $a$  and  $b$  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:

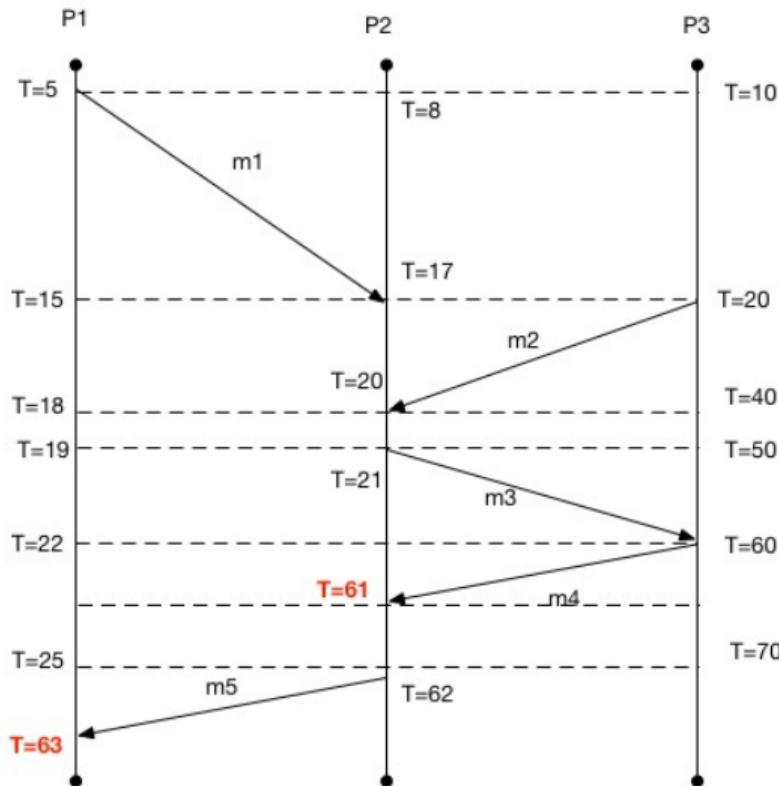
we cannot say whether  $x \rightarrow y$  or  $y \rightarrow x$  from the perspective of the first set of processes

# Example



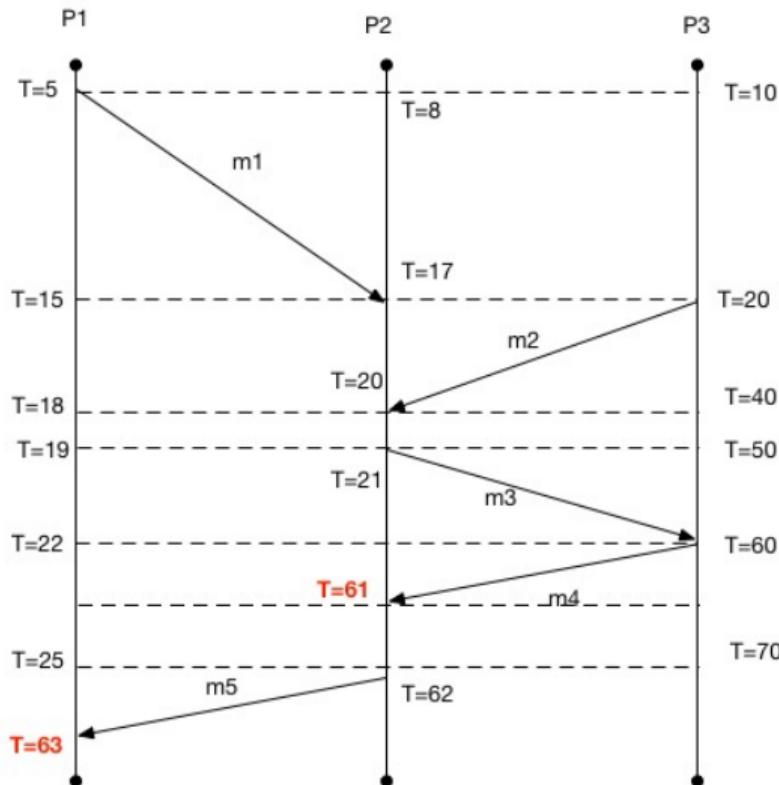
- When a message is transmitted from  $P_1 \rightarrow P_2$ ,  $P_1$  will encode the send time into the message

# Example



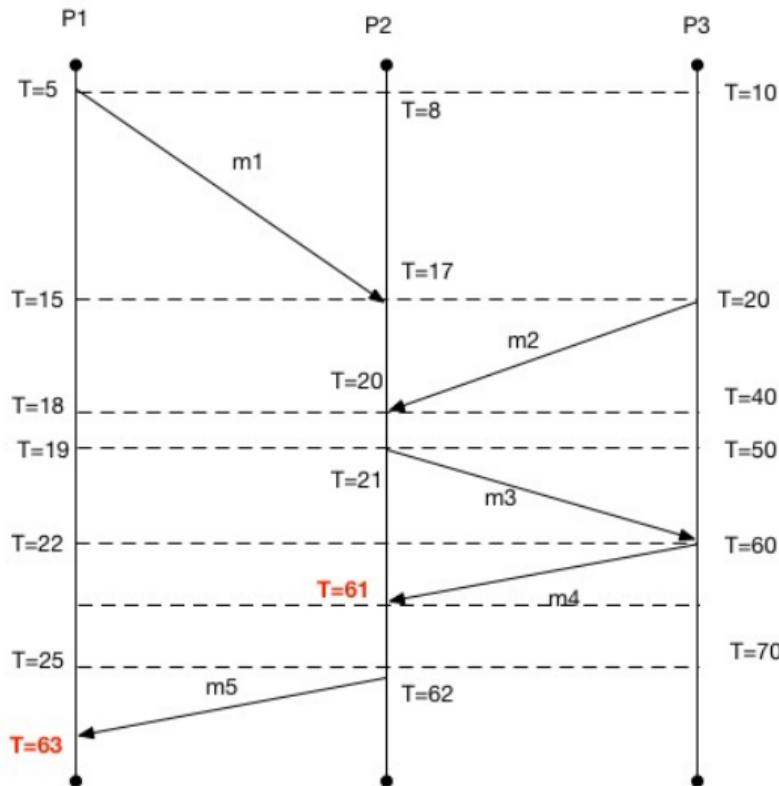
- When a message is transmitted from  $P1 \rightarrow P2$ ,  $P1$  will encode the send time into the message
- When  $P2$  receives the message, it will record the time of receipt

# Example



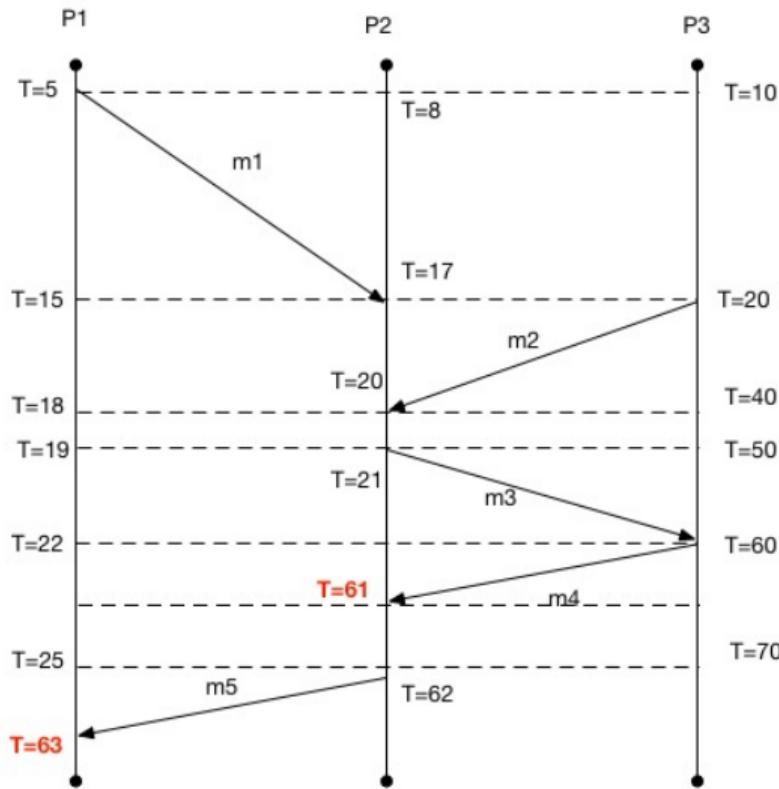
- When a message is transmitted from  $P1 \rightarrow P2$ ,  $P1$  will encode the send time into the message
- When  $P2$  receives the message, it will record the time of receipt
- If the time at  $P2$  is already greater than the send time, then no action is required for  $P2$

# 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

# Example

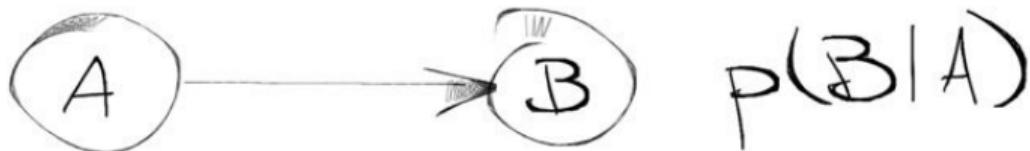


- When a message is transmitted from  $P1 \rightarrow P2$ ,  $P1$  will encode the send time into the message
- When  $P2$  receives the message, it will record the time of receipt
- If the time at  $P2$  is already greater than the send time, then no action is required for  $P2$
- If  $P2$  discovers 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

# Lamport Clocks Limitations

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

# Causality

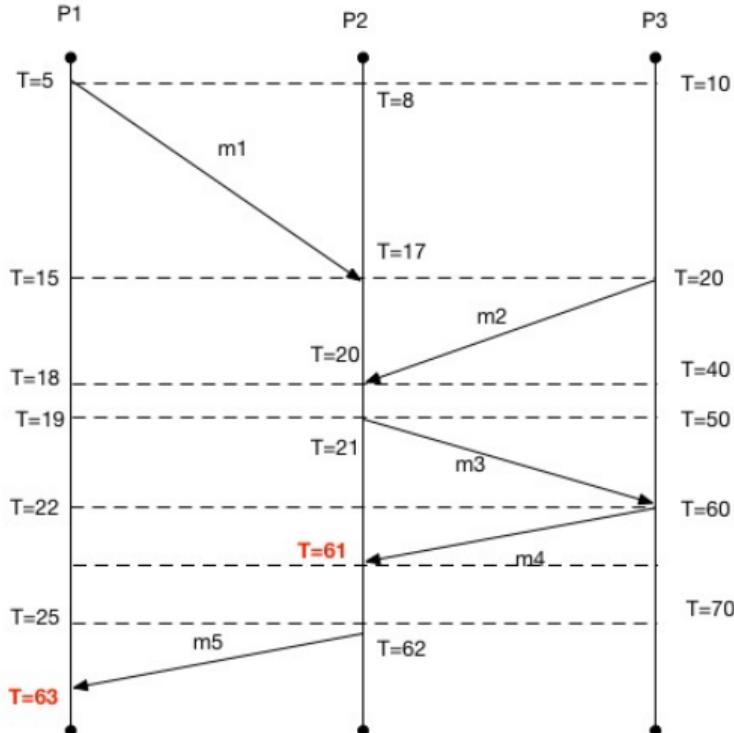


Causal relations

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

Credits: C. Giarmatzi

# Who did when?



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

# Vector Clocks

## Vector Clocks

by Colin Fidge and Friedemann Mattern in 1988

# Vector Clocks

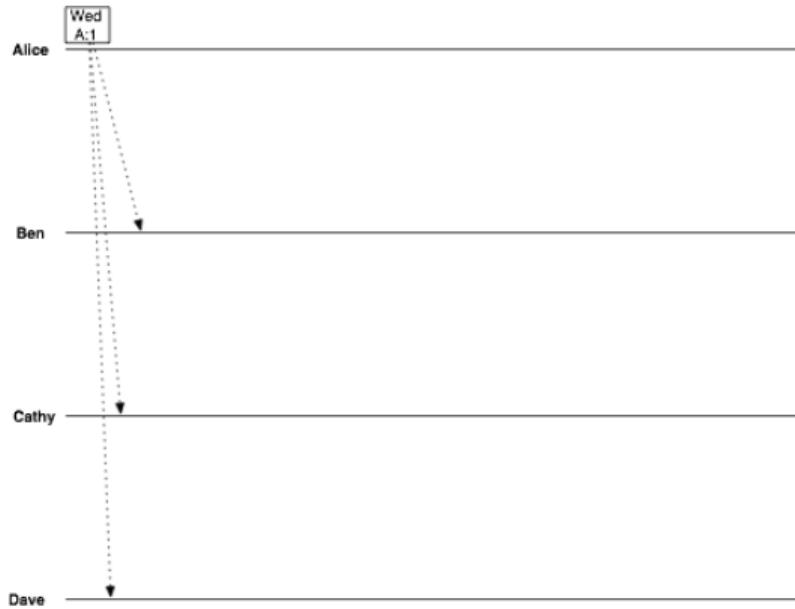
- A vector clock of a system of N processes is **an array/vector** of N logical clocks, **one clock per process**
- A vector clock  $VC(a)$  is assigned to an event  $a$
- If  $VC(a) < VC(b)$  for events  $a$  and  $b$ , then event  $a$  is known to causally precede  $b$

# Robbery

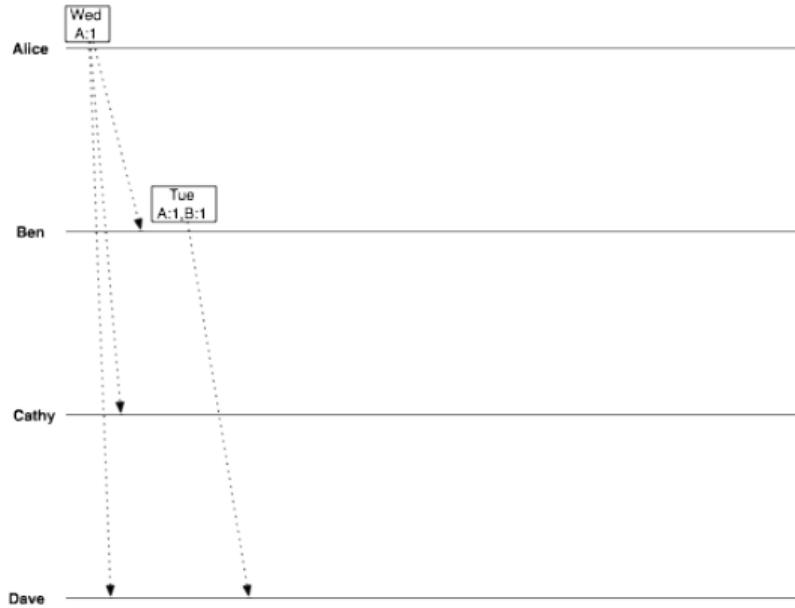


Alice, Ben, Cathy, and Dave want to rob a bank

- They could send P2P messages
- No leader



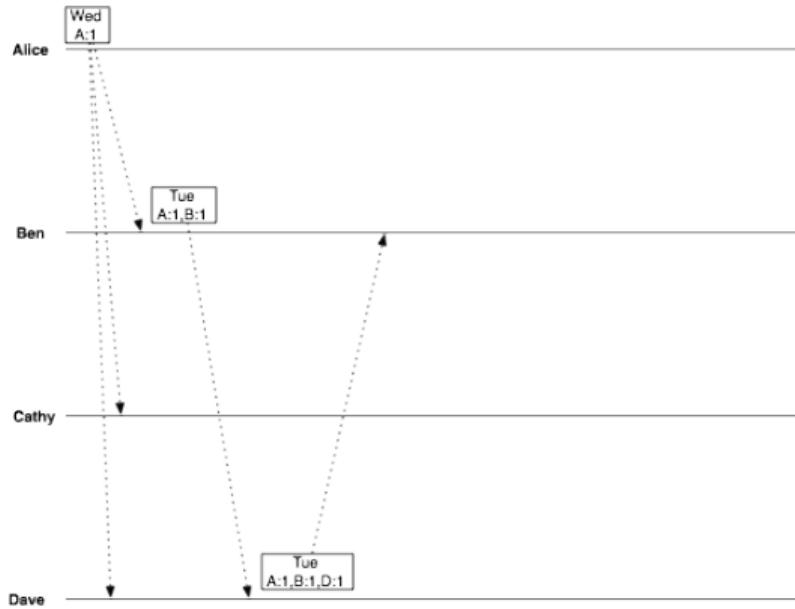
```
date = Wednesday  
vclock = Alice:1
```



Ben suggests Tuesday

date = Tuesday

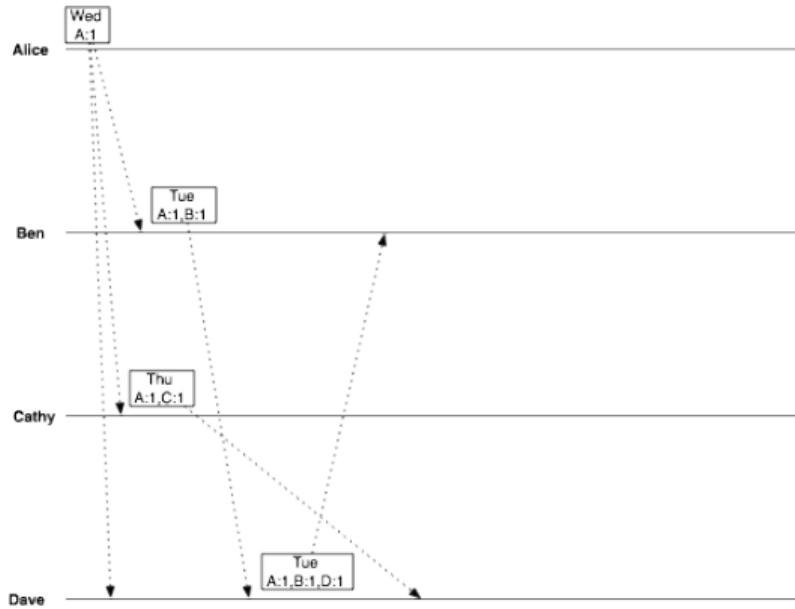
vclock = Alice:1, Ben:1



Dave replies, confirming Tuesday

`date = Tuesday`

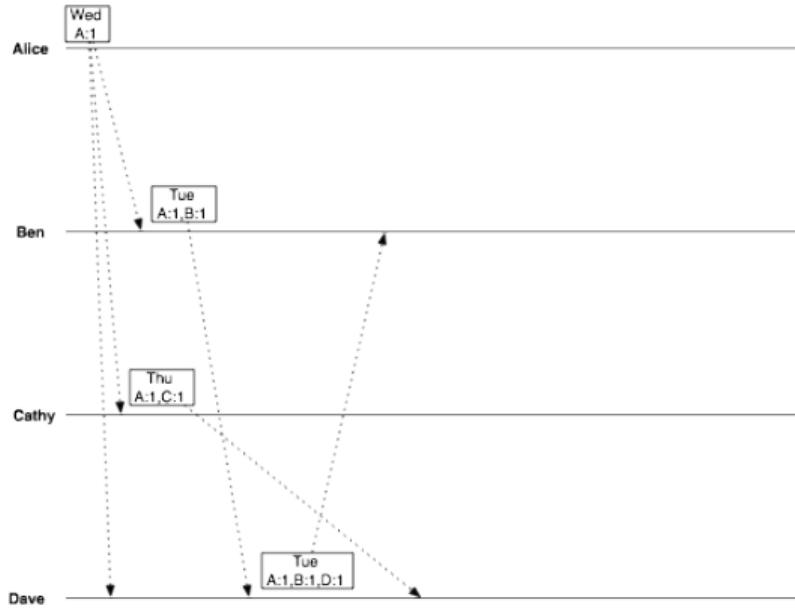
`vclock = Alice:1, Ben:1, Dave:1`



Then Cathy joins, suggesting Thursday

`date = Thursday`

`vclock = Alice:1, Cathy:1`



Dave has two conflicting objects

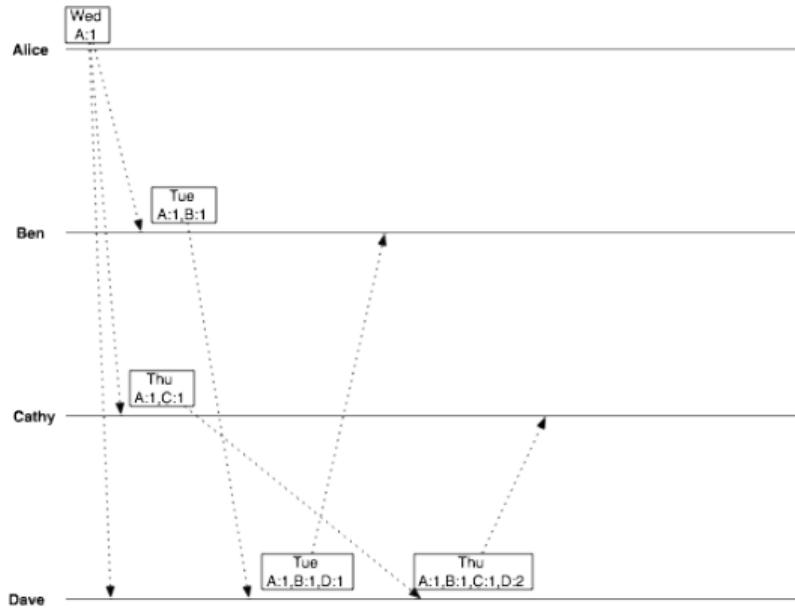
`date = Tuesday`

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

and

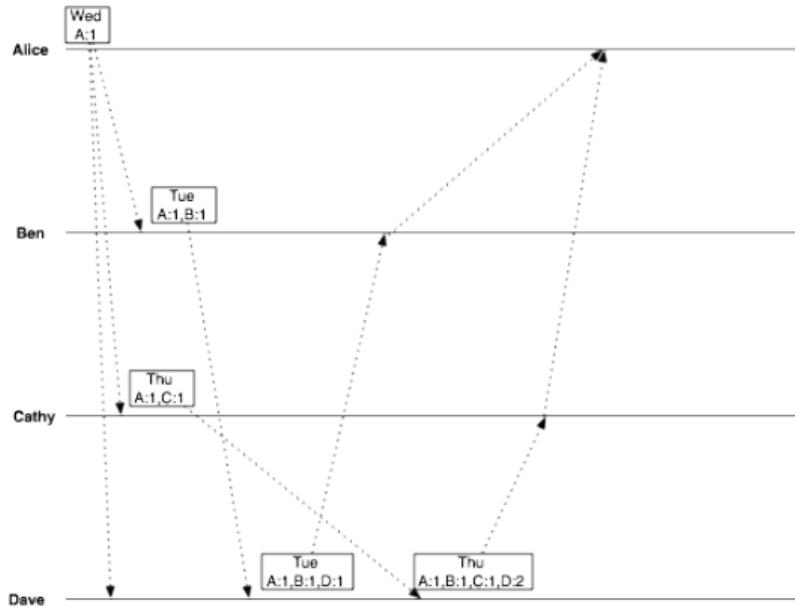
`date = Thursday`

`vclock = Alice:1, Cathy:1`



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
```

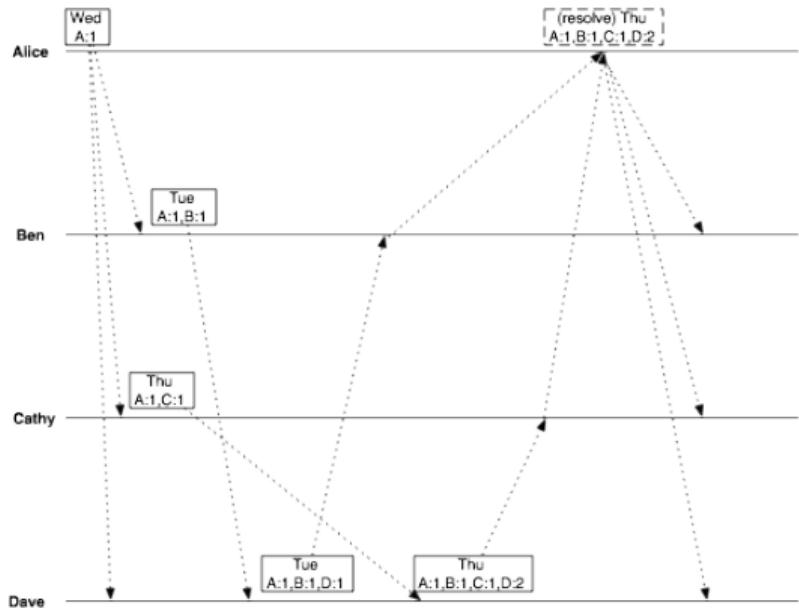


When Alice asks Ben and Cathy for the latest decision, the replies she receives are, from Ben

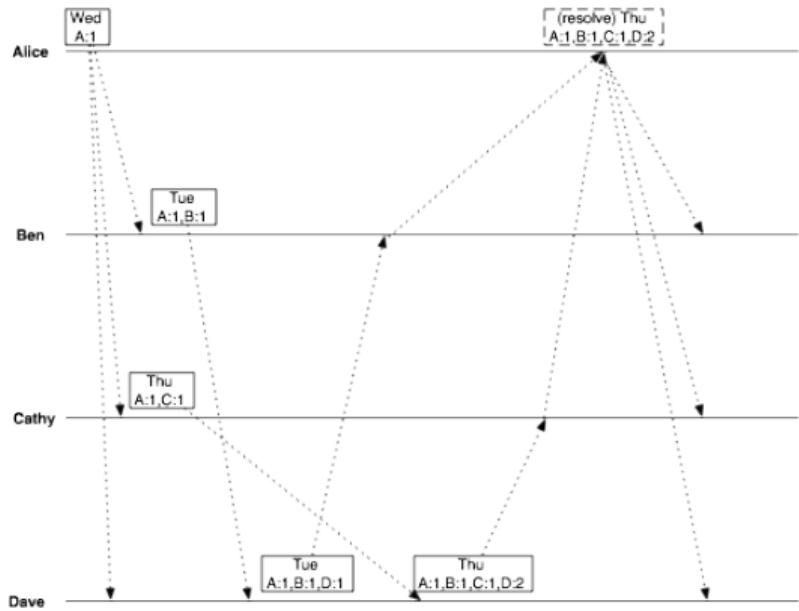
```
date = Tuesday
vclock = Alice:1, Ben:1, Dave:1
```

From Cathy and Dave

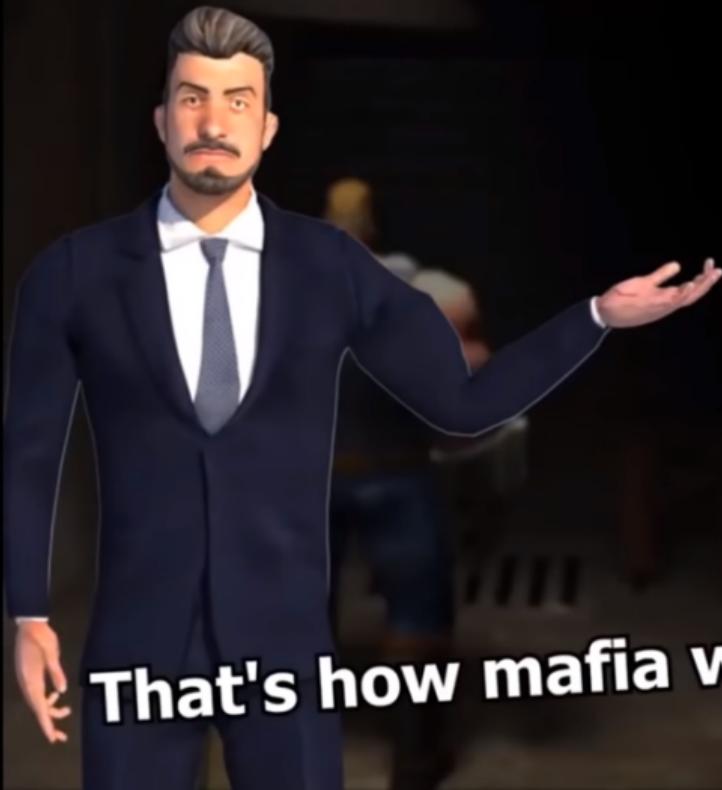
```
date = Thursday
vclock = Alice:1, Ben:1, Cathy:1,
          Dave:2
```



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



**That's how mafia works**

# So, What!?

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

# 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

# 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 were no December 30 in Samoa?

# Thank you! That was Algebraic! Questions?

- Thank
- You
- For
- Your
- Time

