Ordering the Chaos

CONTACT@ADAMFURMANEK.PL

HTTP://BLOG.ADAMFURMANEK.PL

FURMANEKADAM

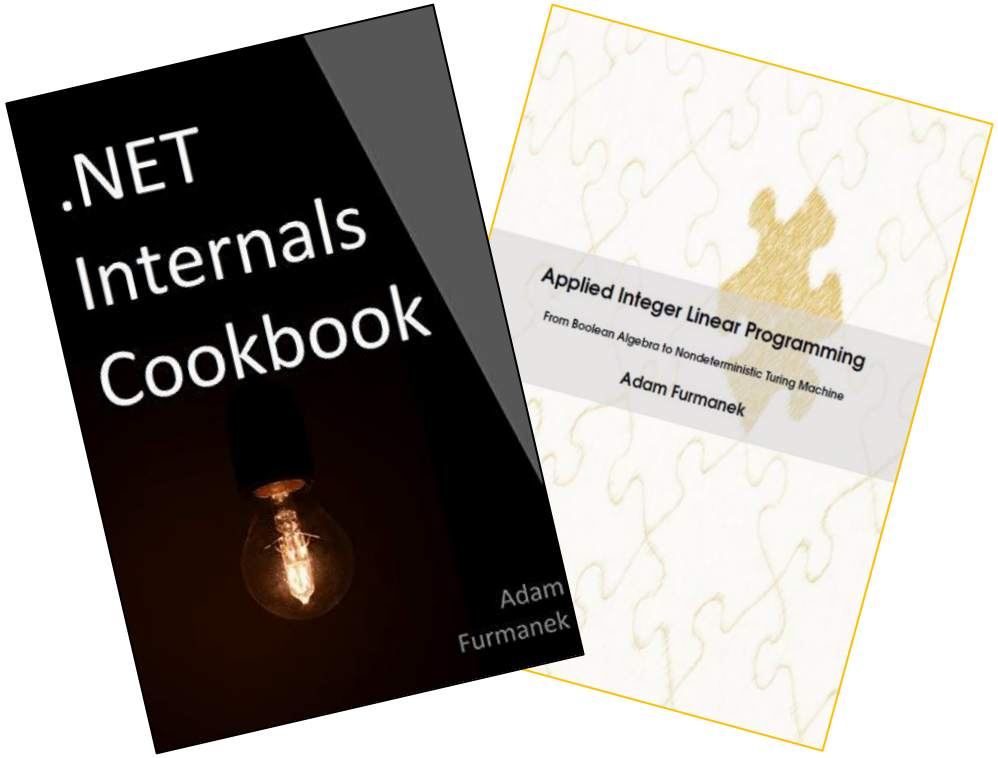
About me

Software Engineer, Blogger, Book Writer, Public Speaker.
Author of ***Applied Integer Linear Programming*** and ***.NET Internals Cookbook***.

http://blog.adamfurmanek.pl

contact@adamfurmanek.pl

furmanekadam



Random IT Utensils

IT, operating systems, maths, and more.

Agenda

What is time?

Using clock in computer science.

Avoiding clock in computer science.

Real implementation.

Going beyond time.

What is time?

What is time

There is no one global time. Each machine has its own clock.

There is a delay between reading the clock value and processing it.

Clocks can differ between readers (Special Theory of Relativity by Einstein).

Clocks break over time (clock drift). Best of them have drift rate around 10^{-13} second.

Standard second is defined as 9,192,631,770 periods of transition between the two hyperfine levels of the ground state of Caesium-133.

Coordinated Universal Time (UTC) is based on atomic time. It is synchronized and broadcasted regularly. Signal can be received with accuracy to about 1 microsecond.

What is timezone

Not (only) a UTC offset!

Region of the globe observing a uniform standard time.

Most of the times it is a whole number of hours offset but can be 30 or 45 minutes.

Specifies offset and Daylight Saving Time (DST) shifts rules.

DST can start at various times of day (2:00 AM, midnight, 0:05 AM) and times of year (as early as March and as late as June).

Storing a time with UTC offset is not enough because the offset may change.

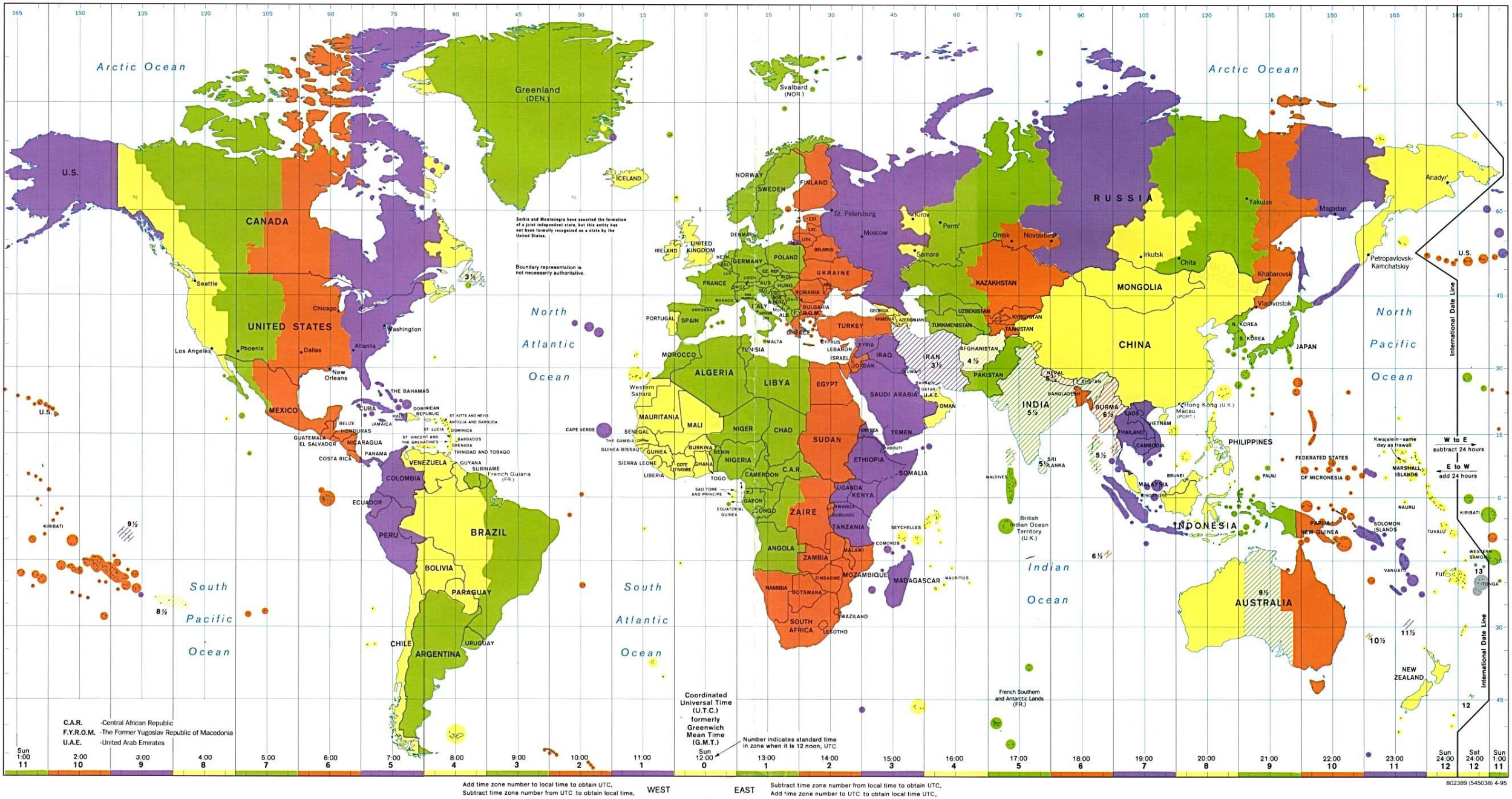
How to show it to user for events half a year from now?

Timezones change

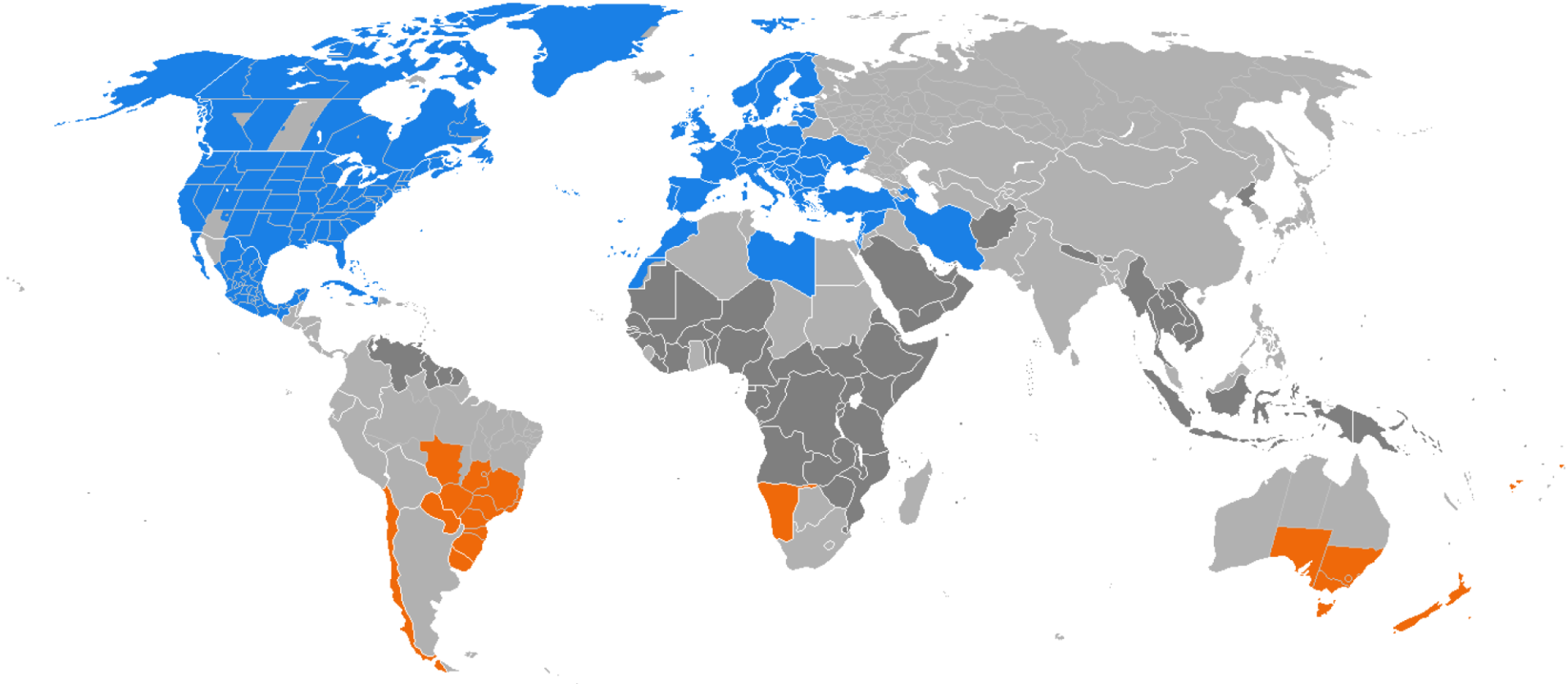
```
Instant date = DatatypeFactory.newInstance().newXMLGregorianCalendar(
    "1899-12-30T07:20:00"
).toGregorianCalendar().toInstant();
System.out.println(LocalDateTime.ofInstant(date, ZoneId.of(
    "Europe/Warsaw"
)).toLocalTime());

// Prints 7:44
```

Timezones for Europe/Warsaw:

- UTC+1h / UTC+2h — since 1977
- UTC+1h — 1965-1976
- UTC+1h / UTC+2h — 1957-1964
- UTC+1:24h — 1800-1914


Standard Time Zones of the World
Arctic Ocean
Greenland (DEN.)
Svalbard (NOR.)
U.S.
CANADA
UNITED STATES
Seattle
Chicago
Washington
Los Angeles
Phoenix
Dallas
Atlanta
New Orleans
MEXICO
THE BAHAMAS
CUBA
DOMINICAN REPUBLIC
HAITI
JAMAICA
BELIZE
HONDURAS
GUATEMALA
EL SALVADOR
NICARAGUA
COSTA RICA
PANAMA
VENEZUELA
GUYANA
SURINAME
French Guiana (FR.)
COLOMBIA
ECUADOR
PERU
BRAZIL
BOLIVIA
PARAGUAY
CHILE
ARGENTINA
URUGUAY
ICELAND
NORWAY
SWEDEN
FINLAND
DENMARK
IRELAND
UNITED KINGDOM
NETH.
BELG.
LUX.
GERMANY
POLAND
CZ. REP.
SLOV.
AUS.
HUNG.
SWITZ.
FRANCE
ROMANIA
MOLD.
UKRAINE
BULGARIA
ALB.
SERB.
MONACO
ANDORRA
ITALY
PORTUGAL
SPAIN
GREECE
MALTA
CYPRUS
TURKEY
MOROCCO
TUNISIA
ALGERIA
LIBYA
EGYPT
Western Sahara
MAURITANIA
MALI
NIGER
CHAD
SUDAN
CAPE VERDE
SENEGAL
THE GAMBIA
GUINEA-BISSAU
GUINEA
BURKINA
BENIN
NIGERIA
SIERRA LEONE
LIBERIA
COTE D'IVOIRE
GHANA
TOGO
CAMEROON
C.A.R.
SAO TOME AND PRINCIPE
GABON
EQUATORIAL GUINEA
CONGO
ZAIRE
UGANDA
KENYA
RWANDA
BURUNDI
TANZANIA
ANGOLA
ZAMBIA
MALAWI
ZIMBABWE
NAMIBIA
BOTSWANA
SOUTH AFRICA
SWAZILAND
LESOTHO
MOZAMBIQUE
MADAGASCAR
MAURITIUS
SEYCHELLES
COMOROS
ERITREA
DJIBOUTI
ETHIOPIA
SOMALIA
YEMEN
SAUDI ARABIA
OMAN
U.A.E.
QATAR
BAHRAIN
KUWAIT
IRAQ
IRAN
3½
JORDAN
LEBANON
ISRAEL
SYRIA
AFGHANISTAN
4½
PAKISTAN
RUSSIA
St. Petersburg
Moscow
Kirov
Perm'
Samara
Omsk
Novosibirsk
Yakutsk
Magadan
Anadyr'
Irkutsk
Chita
Khabarovsk
Vladivostok
Petropavlovsk-Kamchatskiy
KAZAKHSTAN
MONGOLIA
UZBEKISTAN
TURKMENISTAN
KYRGYSTAN
TAJIKISTAN
CHINA
N. KOREA
S. KOREA
JAPAN
NEPAL
5¾
BHUTAN
BANGLADESH
INDIA
5½
SRI LANKA
MALDIVES
BURMA
6½
LAOS
THAILAND
VIETNAM
CAMBODIA
Hong Kong (U.K.)
Macau (PORT.)
PHILIPPINES
BRUNEI
MALAYSIA
SINGAPORE
INDONESIA
British Indian Ocean Territory (U.K.)
AUSTRALIA
9½
8½
10½
11½
PAPUA NEW GUINEA
FEDERATED STATES OF MICRONESIA
PALAU
NAURU
KIRIBATI
SOLOMON ISLANDS
TUVALU
VANUATU
FIJI
TONGA
WESTERN SAMOA
NEW ZEALAND
Kwajalein—same day as Hawaii
MARSHALL ISLANDS
W to E subtract 24 hours
E to W add 24 hours
International Date Line
North Pacific Ocean
South Pacific Ocean
North Atlantic Ocean
South Atlantic Ocean
Indian Ocean
French Southern and Antarctic Lands (FR.)
Serbia and Montenegro have asserted the formation of a joint independent state, but this entity has not been formally recognized as a state by the United States.
Boundary representation is not necessarily authoritative.
C.A.R. -Central African Republic
F.Y.R.O.M. -The Former Yugoslav Republic of Macedonia
U.A.E. -United Arab Emirates
Coordinated Universal Time (U.T.C.) formerly Greenwich Mean Time (G.M.T.)
Number indicates standard time in zone when it is 12 noon, UTC
Add time zone number to local time to obtain UTC. Subtract time zone number from UTC to obtain local time.
WEST
EAST
Subtract time zone number from local time to obtain UTC. Add time zone number to UTC to obtain local time UTC.
802389 (545038) 4-95

Daylight Saving Time



It goes on and on

Amsterdam was once at the UTC+00:19:32.13 (19 minutes). It was generally simplified to UTC+0:20.

Offsets go from -12 to +14. Kiribati is on the very west of the world but is the first to observe the new year.

There are cities with multiple time zones like Michigan City, IN.

Gold Coast (city in Australia) spans across two states which have different time zones in summer.

Gold Coast Airport's runway crosses the state line. Its one end is in different timezone than the other. Airport observes one time, though.

Russian Railways (except Sakhalin railways) followed Moscow Time. Starting in 2018 they follow local time.

Genrally, Arizona doesn't observe DST. Navajo Reservation in Arizona does, while Hopi Reservation does not. Jeddito, AZ is a town in Navajo Reservation, surrounded by the Hopi Reservation, which is in turn surrounded by the rest of the Navajo Reservation.

UTC vs GMT

UTC

Based on atomic clock.

Is an approximation of GMT.

Uses leap seconds to stay close to GMT.

Is a time.

GMT

Based on rotation of the Earth.

Can differ from UTC by up to 0.9 second.

Now replaced by UT1.

Is a timezone.

Leap second

Second added to UTC time to maintain distance to solar time. It can be deleted but this hasn't happened.

It can break your system! It happened on 2012-06-30.

The Altea reservation and departure system run by Amadeus, one of the largest computer travel reservation systems on the planet, couldn't cope and crashed. For 48 minutes, passengers and staff at Qantas and Virgin Australia were thrown back into the 1990s world of manual check-in and delayed flights.

The problem was (...) Linux, and back then the addition or removal of a leap second sent the system into meltdown – the system would deadlock.

The bug was found to affect kernels version numbers 2.2.26 to 3.3, inclusive.

https://www.theregister.co.uk/2015/01/09/leap_second_bug_linux_hysteria/

How is time distributed

Internet Assigned Numbers Authority (IANA) distributes a database called Time Zone Database (tz or zoneinfo).

It is updated multiple times a year.

For example 2019b released on 2019-07-01.

RFC 6557 Procedures for Maintaining the Time Zone Database describes how to update the time.

Most Linux distributions use *tzdata* package which gets regular updates.

How is time distributed

Unicode Common Locale Data Repository (CLDR) provides mappings for languages, timezones, locales, parsing formats, country codes and much more.

Used by Microsoft (Windows), Apple (iOS), IBM, Google and many more.

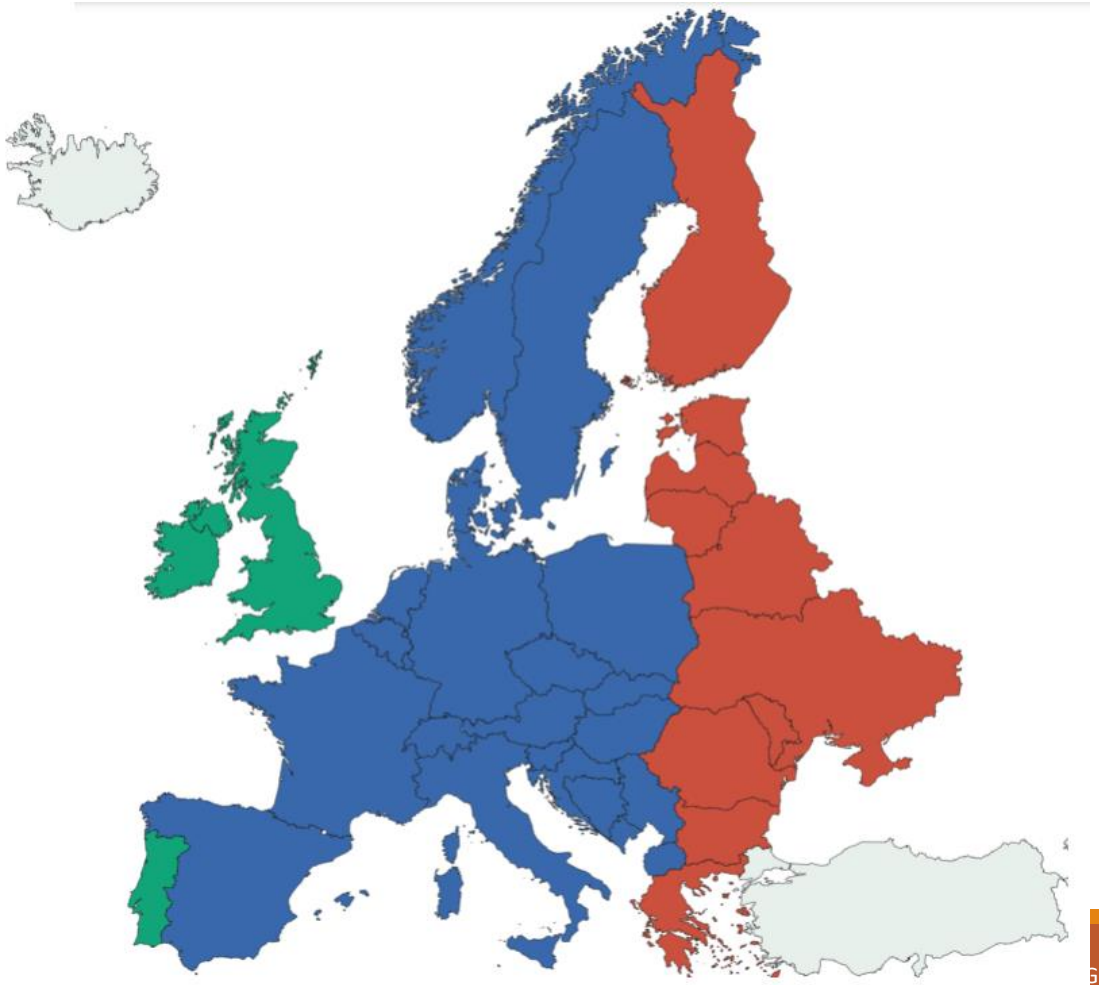
Distributed as XML files.

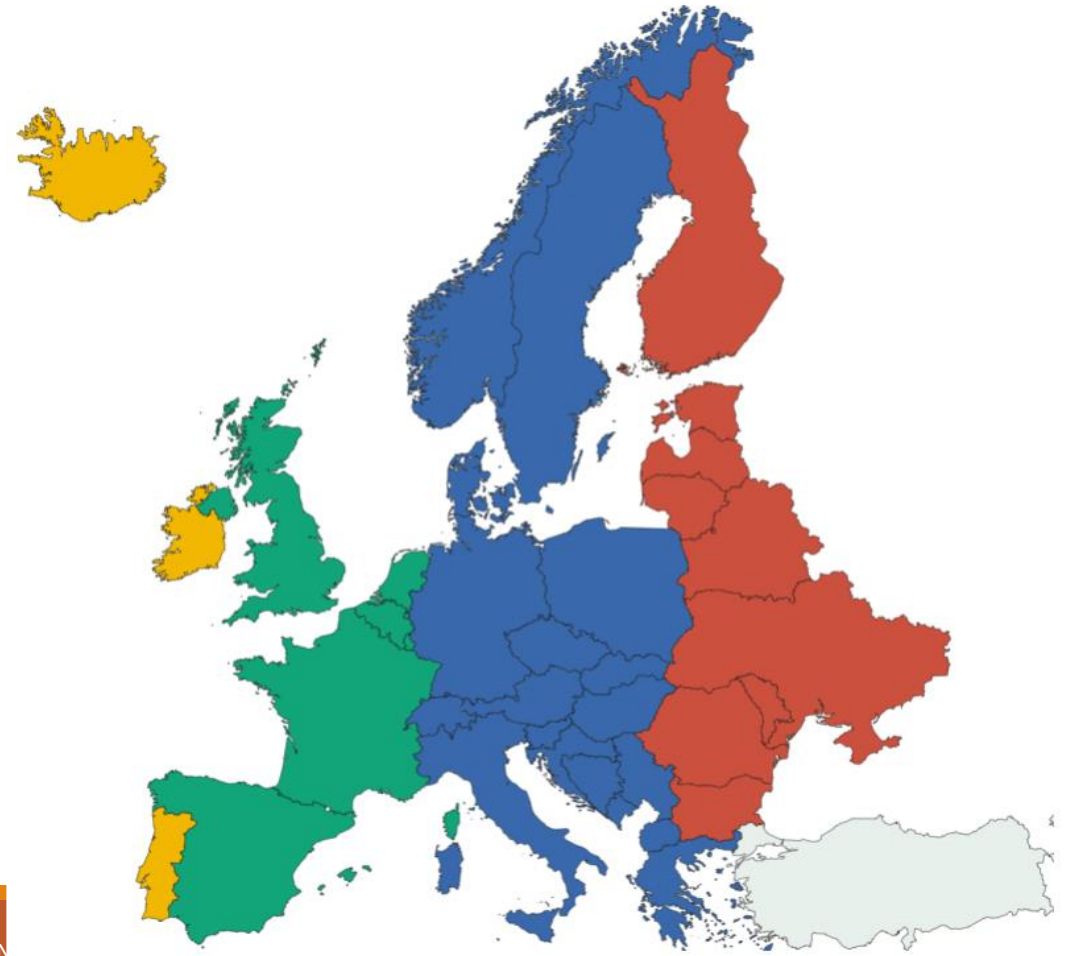
For example version 35.1 released on 2019-04-17.

Local time or UTC?

https://www.euronews.com/my-europe/2022/10/28/when-will-the-eu-end-seasonal-clock-changes-only-time-will-tell

UTC is not a silver bullet





UTC is not a silver bullet

European Union considered dropping DST changes.

We want to organize an event at **9AM** on **May 18th 2025**.

Currently, expected timezone is UTC+2.

Imagine that the country decides to go with UTC+1. This change will be published sometime next year.

How do we store the start time now?

UTC is not a silver bullet
Let’s store as UTC and never update

We can store the value in UTC timezone. We subtract 2 hours from 9AM and get the value:

2025-05-18T**07:00**:00Z

Country changes timezone. User comes to the system, we get UTC time, add 1 hour and get **8AM**. We are one hour ahead of the event!

Pros:

- Easy to implement

Cons:

- Doesn’t work – we have off by one error

UTC is not a silver bullet
Let’s store as UTC and update

We can store the value in UTC timezone. We subtract 2 hours from 9AM and get the value:

2025-05-18T**07:00**:00Z

When we get an update of tz database, we recalculate the time. This time we get:

2025-05-18T**08:00**:00Z

Pros:

- It gives correct result

Cons:

- We need to store the original tz database version along with the data
- We need to access historical data for recalculations

UTC is not a silver bullet
Let’s store as local time

Store date provided by an organizer – **event is at 9AM**.

Whenever we get a request we recalculate the time on the fly.

Pros:

- It works

Cons:

- We need to recalculate every time
- If we cache results then we need to update them when tz database changes

```
News for the tz database

Release 2022b - 2022-08-10 15:38:32 -0700

  Briefly:
    Chile's DST is delayed by a week in September 2022.
    Iran no longer observes DST after 2022.
    Rename Europe/Kiev to Europe/Kyiv.
    New zic -R option
    Vanguard form now uses %z.
    Finish moving duplicate-since-1970 zones to 'backzone'.
    New build option PACKRATLIST
    New tailored_tarballs target, replacing rearguard_tarballs

  Changes to future timestamps

    Chile's 2022 DST start is delayed from September 4 to September 11.
    (Thanks to Juan Correa.)

    Iran plans to stop observing DST permanently, after it falls back
    on 2022-09-21.  (Thanks to Ali Mirjamali.)
```

https://data.iana.org/time-zones/tzdb-2022b/NEWS

Problems with time

Minute has 60 seconds.

Month starts and ends on the same year.

Year has 365 days.

February has 28 days.

Week begins and ends in the same month.

Leap second is always inserted (never deleted).

Timezone is a whole number of hours offset.

https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time

A month begins and ends in the same year

Country	Start numbered year on 1 January	Adoption of Gregorian Calendar
France	1564	1582
Poland	1556	1582
Russia	1700	1918
Scotland	1600	1752
Spain	1556	1582
Sweden	1559	1753
Venice	1797	1582

https://en.wikipedia.org/wiki/Gregorian_calendar#Beginning_of_the_year

February has 28 days

Every 4 years (more or less) it has 29 days.

It can have 30 days. It happened for real in Sweden in 1712.

In 1753 February 17 was followed by March 1.

Not to mention Symmetry454 calendar containing a 35-days February.

A minute lasts 60 seconds or something like that. Definitely not an hour!

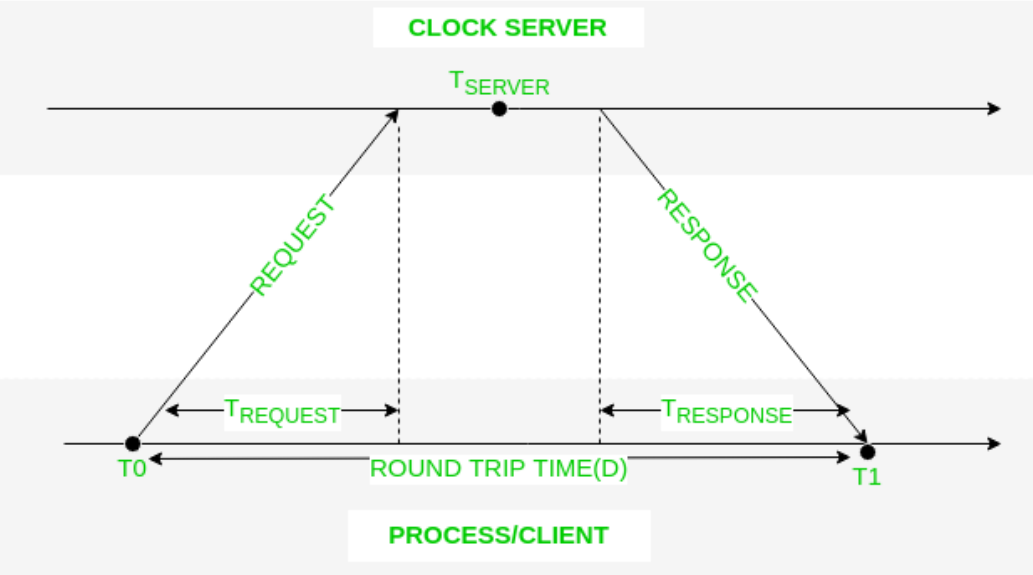
Due to a bug in KVM on CentOS a virtual machine didn’t update its time when the system was put to sleep.

Whenever you suspended your machine its clock was drifting away. This could last minutes, hours or days.

https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time

Using clock in computer science

Cristian's algorithm for clock synchronization



We send request to server at time T_0 and get answer at time T_1.

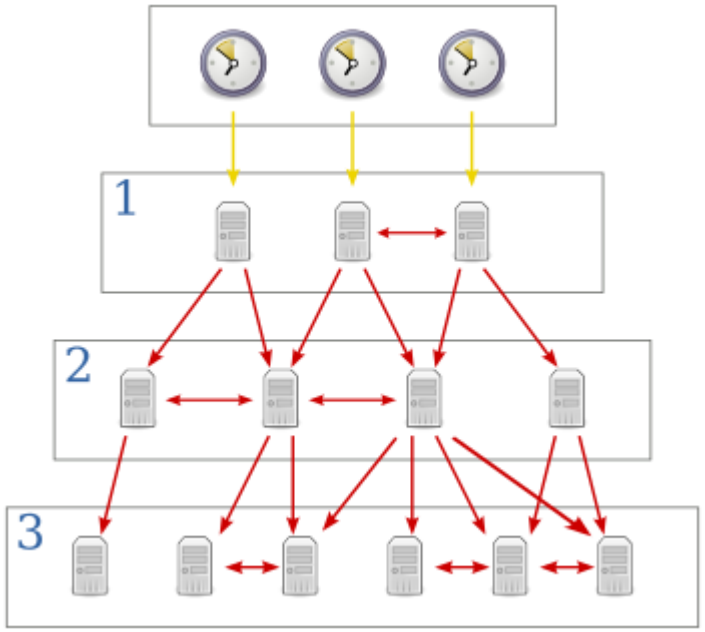
We set the time to $T_{client} = T_{server} + \frac{(T_1 - T_0)}{2}$

This bounds the error.

We repeat the process multiple times and choose response with lowest round trip time.

https://www.geeksforgeeks.org/cristians-algorithm/

Network Time Protocol (NTP)



We group machines in so called STRATUM layers.

STRATUM 0 is based on atomic clocks.

STRATUM 1 is synchronized within few microseconds.

There are multiple versions of standard. NTPv4 passes 128-bit timestamps.

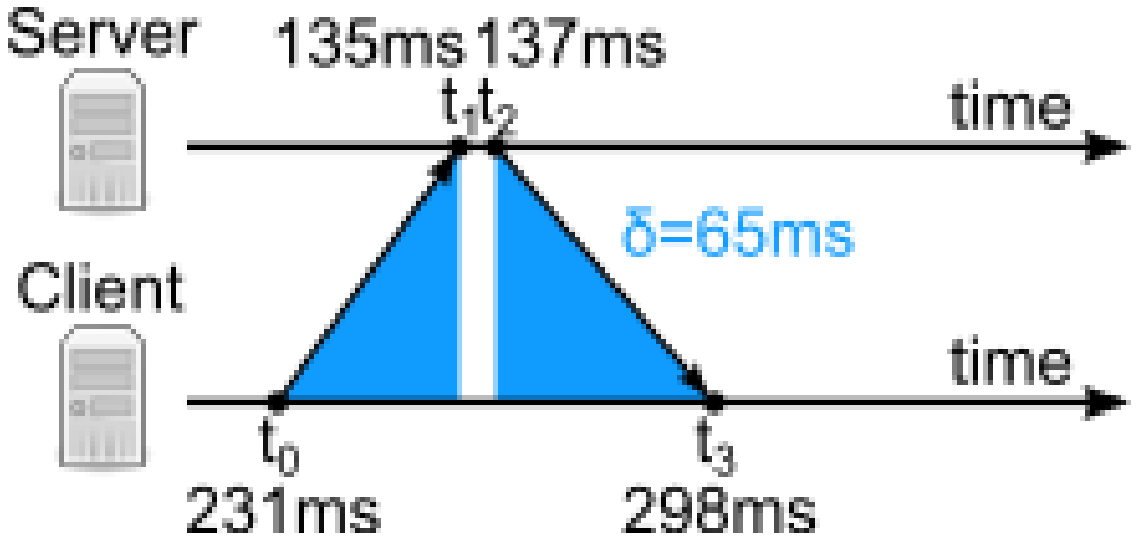
Network Time Protocol (NTP)

Client polls multiple servers and performs statistical analysis.

It calculates:

- time offset $\theta = \frac{(t_1 - t_0) + (t_2 - t_3)}{2}$
- round-trip delay $\delta = (t_3 - t_0) - (t_2 - t_1)$

Outliers are discarded and time is **estimated**.



Other approaches

Marzullo's algorithm

- Estimates accurate time based on noisy sources

Intersection algorithm

- Used by NPT
- Similar to Marzullo's algorithm, calculates center of interval differently

TrueTime

- Used by Google to synchronize time
- Each timestamp has a confidence interval no longer than 7ms
- Spanner utilizes timestamps to order transactions

Avoiding clock in computer science

Clock we want

We want to be able to answer if event a happened before event b.

We want to do it on multiple machines over the internet.

We want it to be fast, we can't wait for miliseconds.

We are interested only in events of some flow — HTTP request, offline job execution etc.

Lamport timestamp

Tells whether event a influenced b which we denote as $a \rightarrow b$.

Provides partial ordering of events across distributed system.

Logical clock counter maintained in each process separately.

Clock increases with every action within single process.

Across processes clock is synchronized when comunication is performed. **Maximum of two values is chosen**.

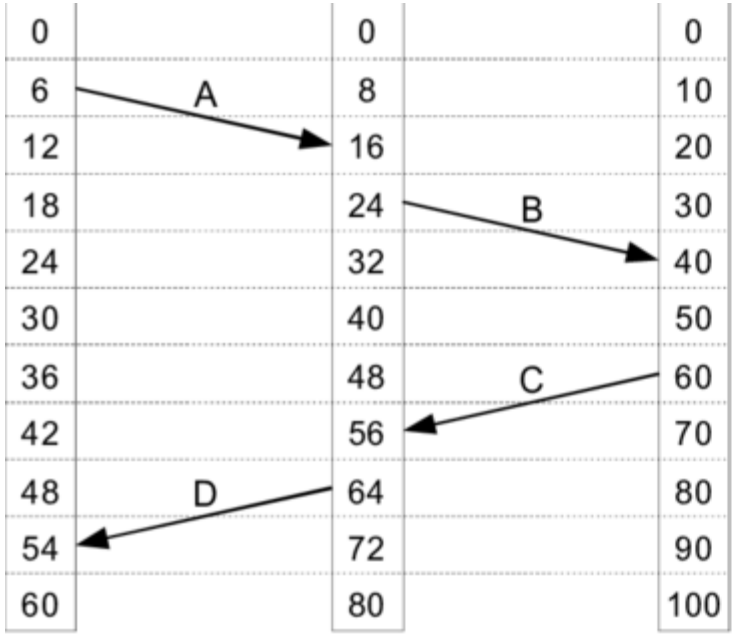
Lamport timestamp

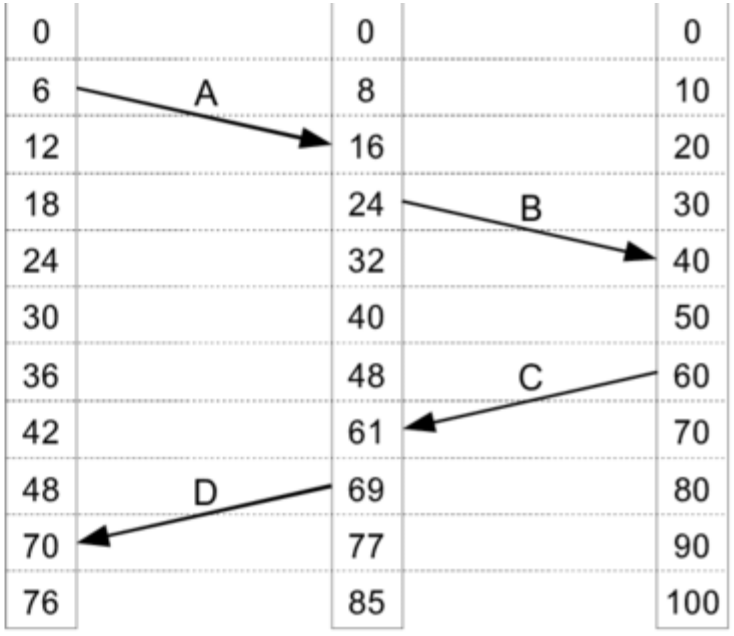
1. A process increments its counter before each event in that process.

2. When a process sends a message, it includes its counter value with the message.

3. On receiving a message, the counter of the recipient is updated, if necessary, to the greater of its current counter and the timestamp in the received message. The counter is then incremented by 1 before the message is considered received.

Lamport timestamp





https://www.researchgate.net/figure/Lamport-timestamps-a-Three-processes-each-with-its-own-clock-The-clocks-run-at_fig7_246857366

Lamport timestamp

If event a happened before b and a influenced b ($a \rightarrow b$) then $C(a) < C(b)$.

It works only when we can guarantee that one event influenced another. It holds within the same machine or across communicating machines.

Knowing that $a \rightarrow c$ and $b \rightarrow c$ we know that c didn't cause a or b but we don't know which initiated c.

Real implementation

Let’s start with manual implementation first and then examine standards.

Implementation

CORRELATION ID

Unique for given logical flow (i.e. User request)

Maintained across all involved parties.

Generated when message comes into the system.

Never modified.

LOGICAL TIME

Local to machine (thread, core, fiber…).

Updated in communication points.

Ideally, passed automatically throughout the system.

Correlator

```
namespace DomainCommons.Correlations
{
    public interface ICorrelator
    {
        string GetCorrelationId();
        int GetLogicalTime();
        void UpdateLogicalTime(int newTime);
        string Activity { get; set; }
    }
}
```

```
using System;
using System.Threading;

namespace DomainCommons.Correlations
{
    public abstract class Correlator : ICorrelator
    {
        private int logicalTime;

        public abstract string GetCorrelationId();
        public abstract string Activity { get; set; }

        public int GetLogicalTime()
        {
            return Interlocked.Increment(ref logicalTime);
        }

        public void UpdateLogicalTime(int newTime)
        {
            int currentTime, finalTime;

            do
            {
                currentTime = logicalTime;
                finalTime = Math.Max(currentTime, newTime);
            } while (Interlocked.CompareExchange(ref logicalTime, finalTime, currentTime)
            != currentTime);
        }
    }
}
```

Correlator

```
package pl.adamfurmanek.correlations.domaincommons;

public interface Correlator {
    String getCorrelationId();
    int getLogicalTime();
    void updateLogicalTime(int newTime);
    String getActivity();
    void setActivity();
}
```

```java
package pl.adamfurmanek.correlations.domaincommons;

public class BaseCorrelator implements Correlator {
    private AtomicInteger logicalTime = new AtomicInteger();

    public abstract String getCorrelationId();
    public abstract String getActivity();
    public abstract void setActivity();

    public int getLogicalTime() {
        return logicalTime.incrementAndGet();
    }

    public void updateLogicalTime(int newTime) {
        int currentTime, finalTime;

        do {
            currentTime = logicalTime.get();
            finalTime = Math.max(currentTime, newTime);
        } while(logicalTime.compareAndSet(currentTime, finalTime);
    }
}
```

Logger

```
namespace DomainCommons.Loggers
{
    public interface ILogger
    {
        void Log(LogLevel level, string message);
        ICorrelator Correlator { get; }
    }
}
```

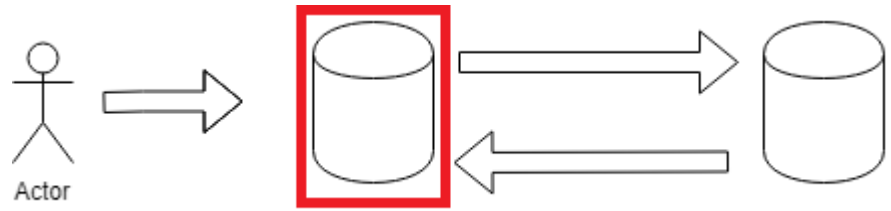
```
using System;
using System.Globalization;
using System.Threading;
using DomainCommons.Correlations;

namespace DomainCommons.Loggers
{
    public abstract class Logger : ILogger
    {
        private readonly string loggerId;

        public Logger(ICorrelator correlator)
        {
            Correlator = correlator;
            loggerId = Guid.NewGuid().ToString();
        }

        public void Log(LogLevel level, string message)
        {
            var segments = new object[]
            {
                Timestamp,
                ApplicationName,
                InstanceId,
                Thread.CurrentThread.ManagedThreadId,
                Correlator.GetCorrelationId(),
                level,
                Correlator.Activity,
                Correlator.GetLogicalTime(),
                loggerId
            };
            LogWithNewLine($"\n[{string.Join("][", segments)}]\n{message}\n");
        }

        public ICorrelator Correlator { get; }
        protected virtual string Timestamp => DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss.fff", CultureInfo.InvariantCulture);
        protected abstract void LogWithNewLine(string message);
        protected abstract string ApplicationName { get; }
        protected abstract string InstanceId { get; }
    }
}
```

Logger

```
package pl.adamfurmanek.correlations.domaincommons;

public interface Logger {
    void log(LogLevel level, String message);
    Correlator getCorrelator();
}
```

```
package pl.adamfurmanek.correlations.domaincommons;

public abstract class BaseLogger implements Logger {
    private final String loggerId;
    private final Correlator correlator;

    public Logger(Correlator correlator) {
        this.correlator = correlator;
        loggerId = UUID.randomUUID().toString();
    }

    public void log(LogLevel level, String message) {
        logWithNewLine(String.format("[%s][%s][%s][%s][%s][%s][%s][%s][%s] %s\n",
            getTimestamp(),
            getApplicationName(),
            getInstanceId(),
            Thread.currentThread().getId(),
            correlator.getCorrelationId(),
            level,
            correlator.getActivity(),
            correlator.getLogicalTime(),
            loggerId,
            message
        ));
    }

    protected String getTimestamp() {
        return new Timestamp(System.currentTimeMillis()).toString();
    }

    protected abstract void logWithNewLine(string message);
    protected abstract string getApplicationName();
}
```

Step 1



User comes to our system.

We need to generate correlation ID and logical time.

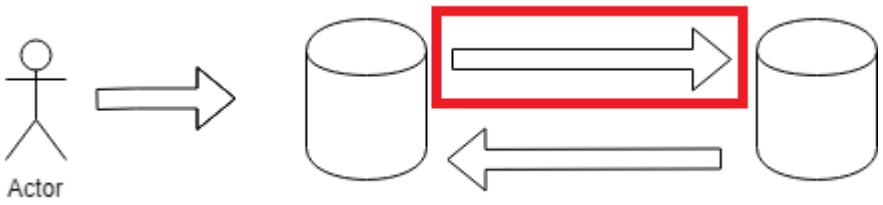
Memory based Correlator

```
using System;
using DomainCommons.Correlations;

namespace AzureCommons.Correlations
{
    public class MemoryBasedCorrelator : Correlator
    {
        private readonly string correlationId = Guid.NewGuid().ToString();

        public override string GetCorrelationId()
        {
            return correlationId;
        }
        public override string Activity { get; set; }
    }
}
```

Memory based Correlator

```
package pl.adamfurmanek.correlations.domaincommons;

public class MemoryBasedCorrelator extends BaseCorrelator {
    private final String correlationId = UUID.randomUUID().toString();

    public String getCorrelationid() {
        return correlationId;
    }
}
```

Step 2



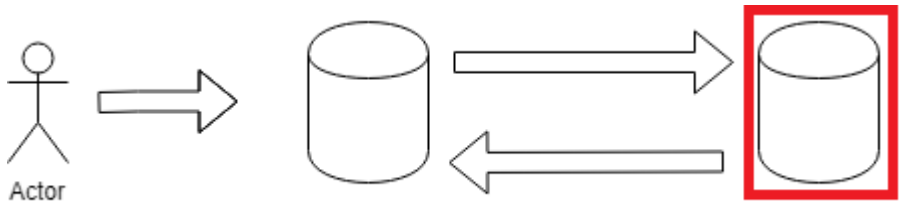
We call some node in the system.

We need to pass correlation ID and logical time in the headers.

```
using System.Linq;
using System.Threading.Tasks;
using DomainCommons.Correlations;
using DomainCommons.Loggers;
using RestSharp;

namespace DomainServices.RestClient
{
    internal class CorrelationRestClient : IRestClient
    {
        private readonly RestSharp.RestClient restClient;
        private readonly ICorrelator correlator;

        public CorrelationRestClient(RestSharp.RestClient restClient, ICorrelator correlator)
        {
            this.restClient = restClient;
            this.correlator = correlator;
        }

        public async Task<IRestResponse<T>> ExecuteTaskAsync<T>(IRestRequest request)
        {
            request.AddHeader(Constants.CorrelationIdHeader, correlator.GetCorrelationId());
            request.AddHeader(Constants.CorrelationCounterHeader, correlator.GetLogicalTime().ToString());

            var result = await restClient.ExecuteTaskAsync<T>(request);

            var correlationHeader = result.Headers.FirstOrDefault(h => h.Name == Constants.CorrelationCounterHeader);
            if (correlationHeader != null)
            {
                correlator.UpdateLogicalTime(int.Parse(correlationHeader.Value.ToString()));
            }

            return result;
        }
    }
}
```

```java
public class CustomClientHeadersFactory implements ClientHeadersFactory {
    @Inject Correlator correlator;

    @Override public MultivaluedMap<String, String> update(
        MultivaluedMap<String, String> incomingHeaders,
        MultivaluedMap<String, String> clientOutgoingHeaders
    ) {
        MultivaluedMap<String, String> returnVal = new MultivaluedHashMap<>();
        returnVal.putAll(clientOutgoingHeaders);

        returnVal.putSingle(Constants.CORRELATION_ID_HEADER, correlator.getCorrelationId());
        returnVal.putSingle(Constants.CORRELATION_COUNTER_HEADER, correlator.getLogicalTime());

        return returnVal;
    }
}
```

Step 3



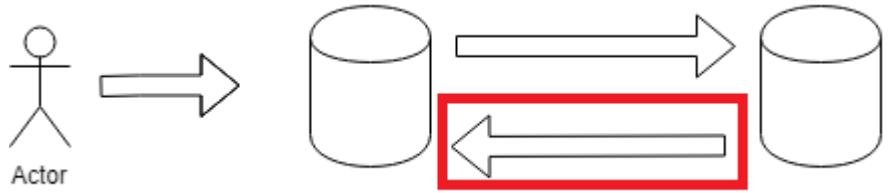
We get HTTP request.

We need to parse headers.

```
using System;
using System.Web;
using DomainCommons.Correlations;
using DomainCommons.Loggers;

namespace AzureCommons.Correlations
{
    public class RequestHeadersCorrelator : Correlator
    {
        private readonly Lazy<string> correlationId;

        public RequestHeadersCorrelator()
        {
            correlationId = new Lazy<string>(GenerateCorrelationId);

            var logicalTime = HttpContext.Current?.Request.Headers[Constants.CorrelationCounterHeader];
            if (logicalTime == null)
            {
                return;
            }

            UpdateLogicalTime(int.Parse(logicalTime));
        }

        public override string GetCorrelationId()
        {
            return correlationId.Value;
        }

        public override string Activity { get; set; }

        private string GenerateCorrelationId()
        {
            var id = (string)HttpContext.Current?.Items[Constants.CorrelationIdItem] ??
                     HttpContext.Current?.Request.Headers[Constants.CorrelationIdHeader] ??
                     Guid.NewGuid().ToString();

            if (HttpContext.Current != null)
            {
                HttpContext.Current.Items[Constants.CorrelationIdItem] = id;
            }

            return id;
        }
    }
}
```

```java
@Component
public class RequestHeaderCorrelator implements Filter Correlator {
    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
      FilterChain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        updateLogicalTime(Integer.parseInt(request.getHeader(Constants.CORRELATION_COUNTER_HEADER)));

        chain.doFilter(req, res);

        HttpServletResponse response = (HttpServletResponse) res;
        response.setHeader(Constants.CORRELATION_COUNTER_HEADER, getLogicalTime().toString());
    }
}
```

Step 4



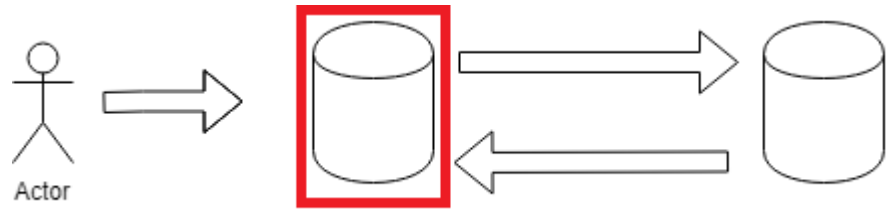
We send response.

We need to return updated logical time in headers.

```
namespace AzureCommons.Filters
{
    public class WebApiCorrelationActionFilterAttribute : ActionFilterAttribute
    {
        public override void OnActionExecuting(HttpActionContext actionContext)
        {
            SetActivity(actionContext);

            var logger = GetFromContainer<ILogger>(actionContext.Request);
            logger.Log(LogLevel.Information, $"Executing with parameters: {string.Join(", ", actionContext.ActionArguments.Select(p => $"{p.Key} = {p.Value}"))}.");
        }

        private static void SetActivity(HttpActionContext actionContext)
        {
            var actionDescriptor = actionContext.ActionDescriptor;
            string actionName = actionDescriptor.ActionName;
            string controllerName = actionDescriptor.ControllerDescriptor.ControllerType.FullName;

            var correlator = GetFromContainer<ICorrelator>(actionContext.Request);
            correlator.Activity = $"{controllerName}.{actionName}";
        }

        public override void OnActionExecuted(HttpActionExecutedContext actionExecutedContext)
        {
            var logger = GetFromContainer<ILogger>(actionExecutedContext.Request);
            logger.Log(LogLevel.Information, "Execution finished");

            var correlator = GetFromContainer<ICorrelator>(actionExecutedContext.Request);
            actionExecutedContext.Response?.Headers?.Add(Constants.CorrelationCounterHeader, correlator.GetLogicalTime().ToString());
            actionExecutedContext.Response?.Headers?.Add(Constants.CorrelationIdHeader, correlator.GetCorrelationId());
        }
    }
}
```

```java
@Component
public class RequestHeaderCorrelator implements Filter Correlator {
    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
      FilterChain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        updateLogicalTime(Integer.parseInt(request.getHeader(Constants.CORRELATION_COUNTER_HEADER)));

        chain.doFilter(req, res);

        HttpServletResponse response = (HttpServletResponse) res;
        response.setHeader(Constants.CORRELATION_COUNTER_HEADER, getLogicalTime().toString());
    }
}
```

Step 5



We need to update headers from the response.

```
using System.Linq;
using System.Threading.Tasks;
using DomainCommons.Correlations;
using DomainCommons.Loggers;
using RestSharp;

namespace DomainServices.RestClient
{
    internal class CorrelationRestClient : IRestClient
    {
        private readonly RestSharp.RestClient restClient;
        private readonly ICorrelator correlator;

        public CorrelationRestClient(RestSharp.RestClient restClient, ICorrelator correlator)
        {
            this.restClient = restClient;
            this.correlator = correlator;
        }

        public async Task<IRestResponse<T>> ExecuteTaskAsync<T>(IRestRequest request)
        {
            request.AddHeader(Constants.CorrelationIdHeader, correlator.GetCorrelationId());
            request.AddHeader(Constants.CorrelationCounterHeader, correlator.GetLogicalTime().ToString());

            var result = await restClient.ExecuteTaskAsync<T>(request);

            var correlationHeader = result.Headers.FirstOrDefault(h => h.Name == Constants.CorrelationCounterHeader);
            if (correlationHeader != null)
            {
                correlator.UpdateLogicalTime(int.Parse(correlationHeader.Value.ToString()));
            }

            return result;
        }
    }
}
```

```java
public class BodyHandler implements MessageBodyReader<Message> {
    @Inject Correlator correlator;

    @Override
    public Message readFrom(Class<Message> type, Type genericType,
                            Annotation[] annotations, MediaType mediaType,
                            MultivaluedMap<String, String> httpHeaders,
                            InputStream entityStream)
            throws IOException, WebApplicationException {

        correlator.updateLogicalTime(Integer.parseInt(httpHeaders.get(Constants.CORRELATION_ID_HEADER)));

        ...
    }

}
```

Important remarks

We want to wire this up using dependency injection or other middleware.

It is crucial to pass Lamport timestamp in each communication method

- Queues
- Database
- Any proprietary RPC framework

Finally, we need to deliver logs to centralized place (Logstash, OMS, Cloud Watch).

Finally, we can just **filter logs using correlation ID** and **sort them using Lamport timestamp**.

```
fields @message
| parse @message "[*][*][*][*][*][*][*][*][*][*] *" as timestamp, appName, instanceId, threadId, correlationId, level, activity, logicalTime, loggerId, message
| filter (correlationId = "00e5fc3b-4f40-4152-baa1-974033de6574")
| display timestamp, appName, level, activity, logicalTime, message
| order logicalTime
```

Run query | Save | History

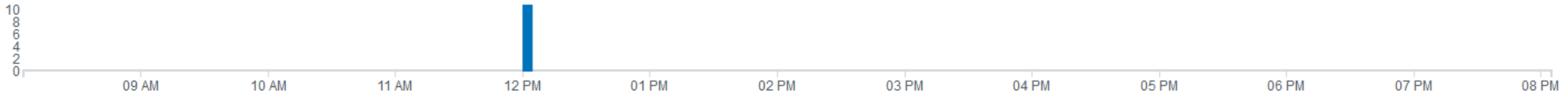
Logs | Visualization

Export results | Add to dashboard

Showing 11 of 11 records matched

25 records (3.9 kB) scanned in 2.4s @ 10 records/s (1.6 kB/s)

Hide histogram



#	timestamp	appName	level	activity	logicalTime	message
▶ 1	2021-08-19T10:00:38.000Z	LoadBalan…	INFO	OpenSession	1	Starting session
▶ 2	2021-08-19T10:00:38.001Z	LoadBalan…	INFO	OpenSession	2	Submitting workflow
▶ 3	2021-08-19T10:00:38.002Z	Payment	INFO	ChargeClient	3	Charging account
▶ 4	2021-08-19T10:00:38.003Z	Mailer	INFO	NotifyOwner	3	Sending emails
▶ 5	2021-08-19T10:00:38.004Z	Mailer	INFO	NotifyOwner	4	Updating calendar
▶ 6	2021-08-19T10:00:38.006Z	Payment	INFO	ChargeClient	4	Finishing payment
▶ 7	2021-08-19T10:00:38.005Z	Mailer	INFO	NotifyOwner	5	Finishing
▶ 8	2021-08-19T10:00:38.008Z	Payment	INFO	ChargeClient	5	Payment accepted
▶ 9	2021-08-19T10:00:38.007Z	Mailer	INFO	NotifyOwner	6	Messages sent
▶ 10	2021-08-19T10:00:38.009Z	LoadBalan…	INFO	OpenSession	7	Finished notifications
▶ 11	2021-08-19T10:00:38.010Z	LoadBalan…	INFO	OpenSession	8	Finished payment

W3C Trace Context

§ *3.2.2.3 trace-id*

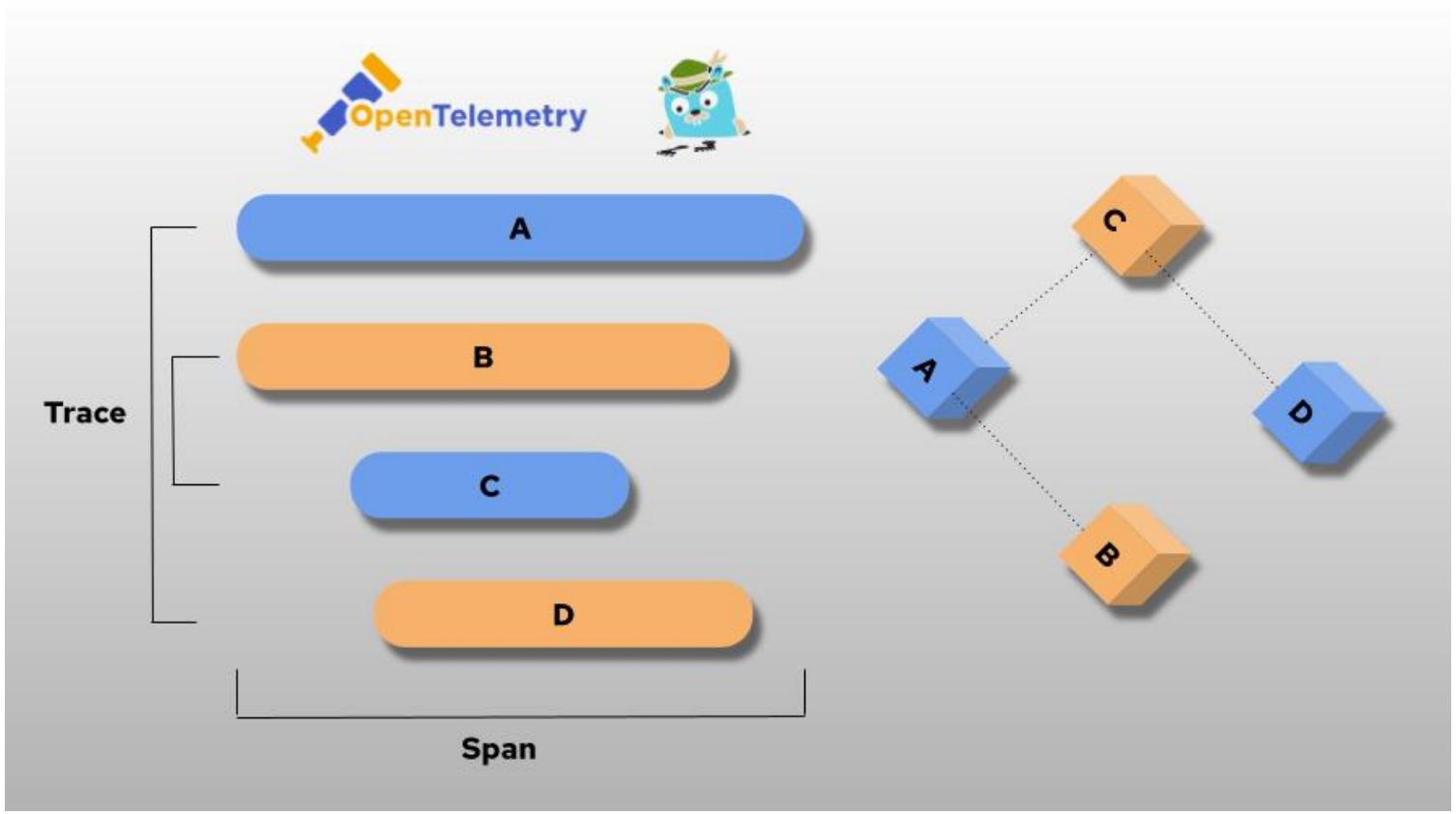
This is the ID of the whole trace forest and is used to uniquely identify a distributed trace through a system. It is represented as a 16-byte array, for example, `4bf92f3577b34da6a3ce929d0e0e4736`. All bytes as zero (`00000000000000000000000000000000`) is considered an invalid value.

If the `trace-id` value is invalid (for example if it contains non-allowed characters or all zeros), vendors *MUST* ignore the `traceparent`.

See considerations for trace-id field generation for recommendations on how to operate with `trace-id`.

https://www.w3.org/TR/trace-context/#trace-id
https://jimmybogard.com/building-end-to-end-diagnostics-and-tracing-a-primer-trace-context/

OpenTelemetry and Jaeger



Out-of-process autoinstrumentation.
No easy way to update the clock on operation return.
Correlation context could be used but is optional.

What's wrong with the image?

https://www.opensourcerers.org/2022/04/18/using-opentelemetry-and-jaeger/

Going beyond time

Vector clock

Generalization of Lamport timestamps.

We have N processes. Each process has its own logical clock.

Each process holds a copy of all clocks and chooses „smallest possible values”.

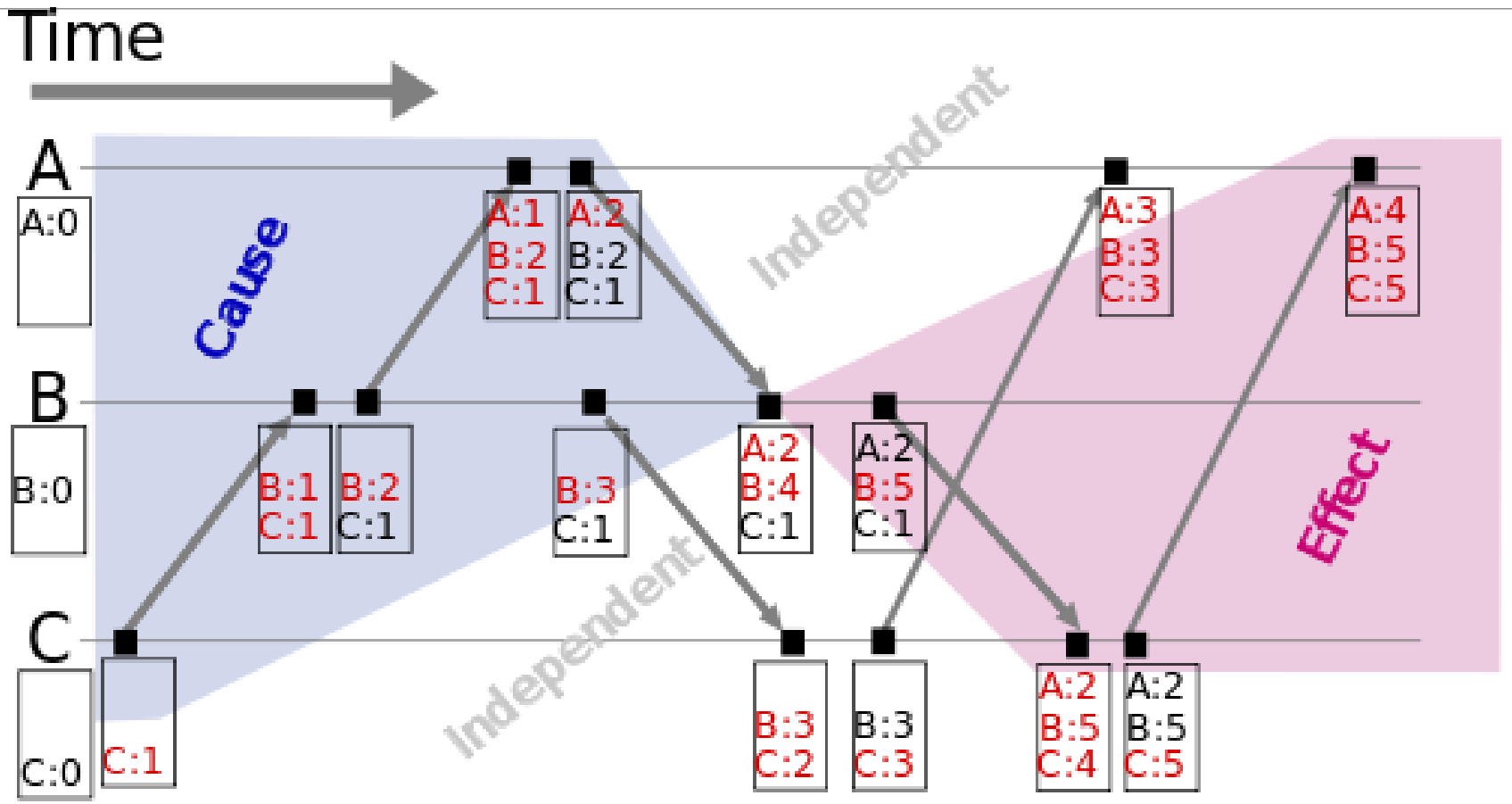
Initially all clocks are zero.

Each time a process experiences an internal event, it increments its own logical clock in the vector by one.

Each time a process sends a message, it increments its own logical clock in the vector by one and then sends a copy of its own vector.

Each time a process receives a message, it increments its own logical clock in the vector by one and updates each element in its vector by taking the maximum of the value in its own vector clock and the value in the vector in the received message (for every element).

Vector clock



https://en.wikipedia.org/wiki/Vector_clock

Vector clock

Provides partial ordering property.

Let's say that $VC(a) = [a_1, a_2, \dots, a_n]$ is a vector clock of a.

We say that $VC(a) < VC(b)$ if for each component $VC(a_i) \leq VC(b_i)$ and for at least one component $VC(a_i) < VC(b_i)$.

If $a \rightarrow b$ then $VC(a) < VC(b)$. Similar to Lamport timestamp.

However, if $VC(a) < VC(b)$ then **we know a happened before b**.

Other clocks

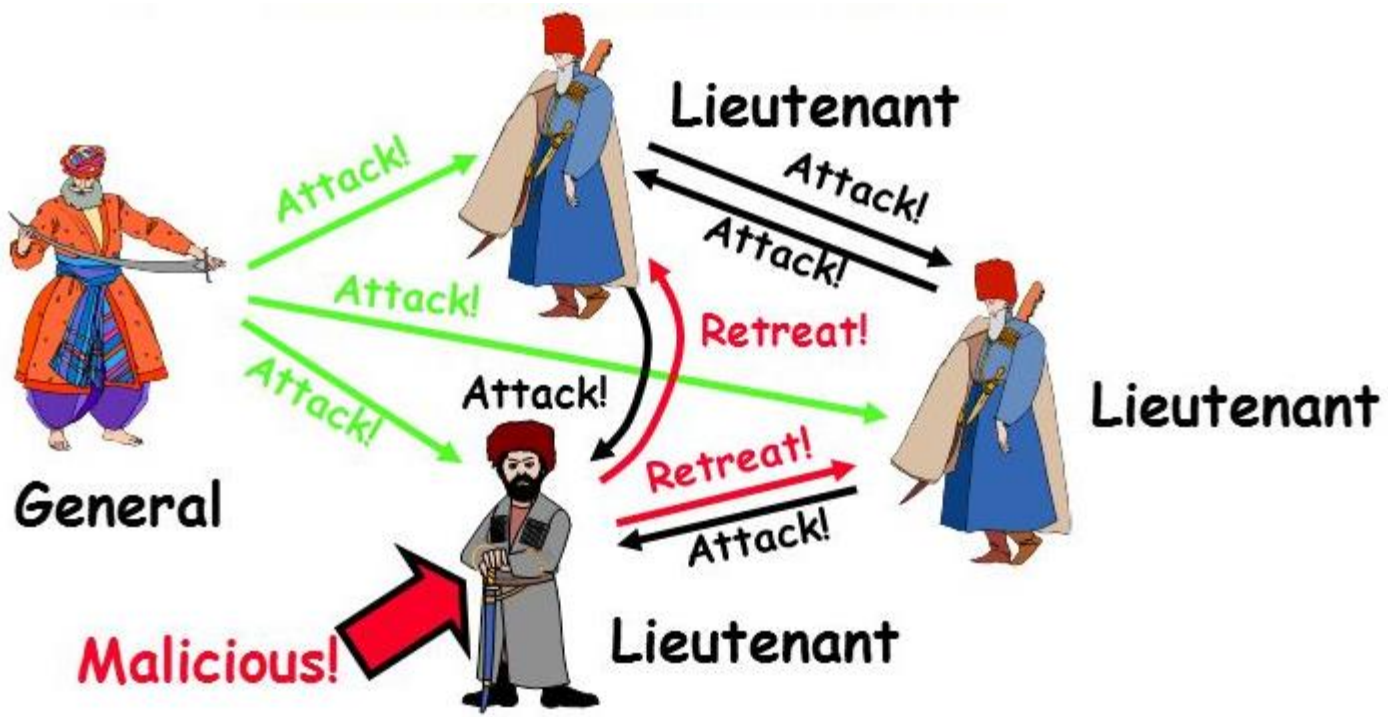
Tree Clocks
- Generalization of vector clocks
- Works when number of processes is dynamic

Plausible Clocks
- Take less space than vector clocks
- Can order events totally

Bloom Clocks
- Probabilistic data structure
- Space complexity independent of the number of nodes in the system
- No false negatives (= if two clocks are not comparable then Bloom Clocks can deduce that)

Matrix clock
- Vector of vector clocks
- Provides lower bounds on what other hosts know

Byzantine generals



Byzantine failure

In distributed systems, component **will fail**.

It may stop responding.

It may violate protocol.

It may repeat messages.

It may send out broken messages.

k-fault tolerance

System is k-fault tolerant if it survives faults in k components and still meets specification.

Without Byzantine failures we need $k + 1$ components to be k-fault tolerant

- We just need to get answer from one component

With Byzantine failures we need $2k + 1$ components to be k-fault tolerant

- We need to do voting with regular majority

Consensus problem

Agreeing on a decision in a distributed system where each node can fail.

We want the following property:

- Termination
  - Every correct process decides some value after a finite steps
- Integrity
  - If all the correct processes proposed the same value then this value must be decided
- Agreement
  - Every correct process must decide on the same value

Consensus problem

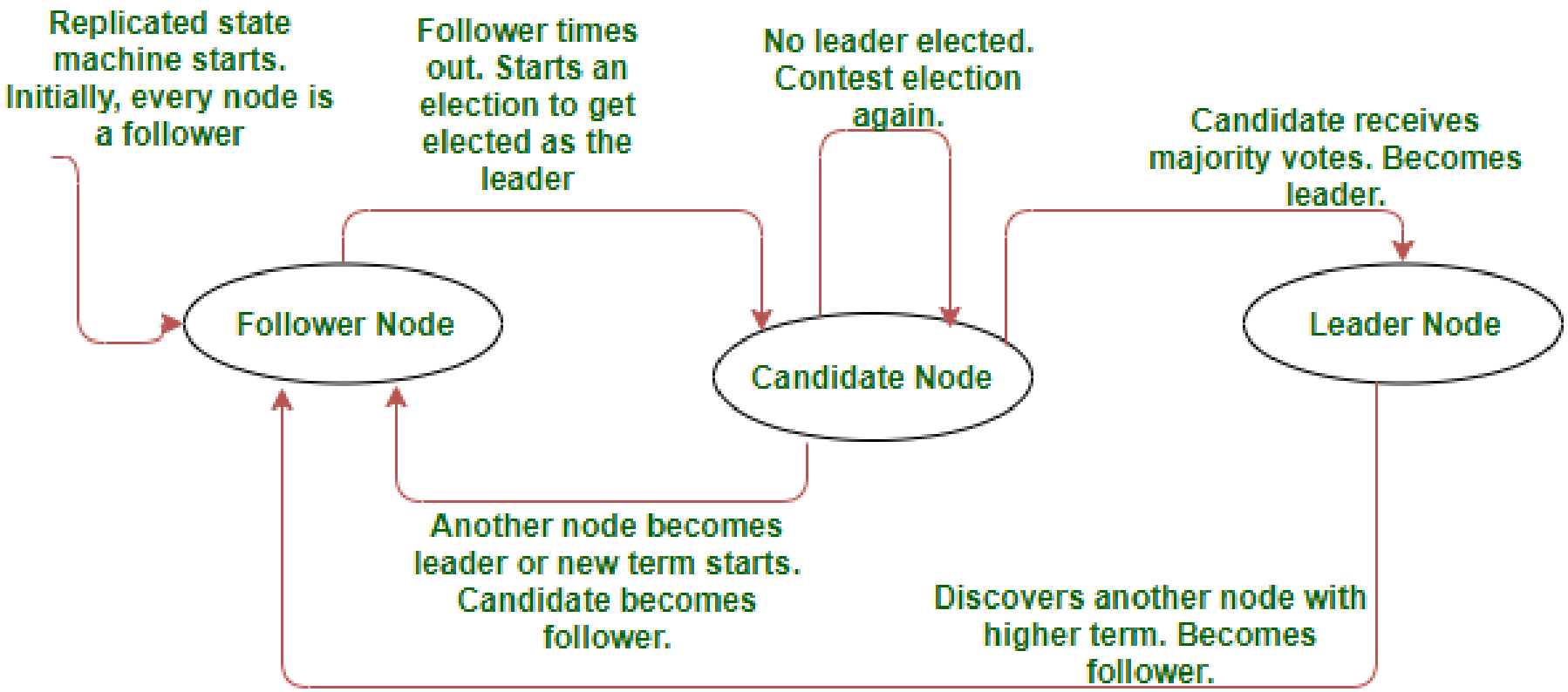
With k faulty components we need $3k + 1$ components in total to reach agreement.

But! If we cannot guarantee bounded message delivery, we cannot reach agreement if one component dies.

Consensus problem

	Unordered		Ordered		
Synchronous			**Yes**		Bounded Delay
			Yes		Unbounded Delay
Asynchronous	**Yes**	**Yes**	**Yes**	**Yes**	Bounded Delay
			Yes	**Yes**	Unbounded Delay
	Unicast	Multicast	Unicast	Multicast	

Raft


Replicated state machine starts. Initially, every node is a follower
Follower times out. Starts an election to get elected as the leader
No leader elected. Contest election again.
Candidate receives majority votes. Becomes leader.
Follower Node
Candidate Node
Leader Node
Another node becomes leader or new term starts. Candidate becomes follower.
Discovers another node with higher term. Becomes follower.

Moving forward

If we have consensus, we can easily implement:

- Total ordered broadcast
- Compare-And-Set (CAS)
- Increment-And-Get (IAG)

Finally, we can easily order logs so we know **exactly** what was happening.

But this introduces very long delays.

This his hard!

https://github.com/jepsen-io/jepsen

A framework for distributed systems verification, with fault injection

Summary

Wall clock is not useful in distributed systems.

Synchronizing clocks is hard. We can do that but we want to avoid doing that.

Logical clocks can be veary simple or very sophisticated. It depends on our needs.

Things will not become easier. Timezones change constantly, we cannot overcome physics limitations, some things are proven to be unsolvable.

Anything in your system can go wrong but if your logging mechanism fails then things are very bad.

Use Jepsen.

Q&A

References

Andres S. Tanenbaum — „Distributed Systems: Principles and Paradigms”

George Coulouris, Jean Dollimore. Tim Kindberg, Gordon Bliar — „Distributed Systems Concepts and Design”

Benjamin Erb — „ Concurrent Programming for scalable web architecture”

Martin Kleppmann — „ Designing Data Intensive Applications”

Brendan Burns — „ Designing Distributed Systems”

Adam Furmanek – „.NET Internals Cookbook”

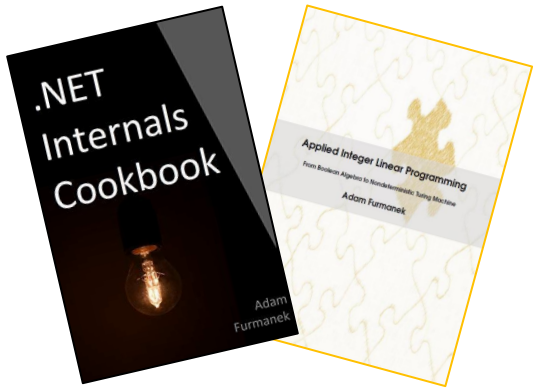
References

https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time — falsehoods programmers believe about time

https://www.researchgate.net/figure/Lamport-timestamps-a-Three-processes-each-with-its-own-clock-The-clocks-run-at_fig7_246857366 — Lamport timestamps

https://medium.com/@balrajasubbiah/lamport-clocks-and-vector-clocks-b713db1890d7 — Lamport clocks and vector clocks

http://blog.adamfurmanek.pl/2017/12/16/logging-in-distributed-system-part-1/ — logging in distributed system implementation



Random IT Utensils

IT, operating systems, maths, and more.

Thanks!

CONTACT@ADAMFURMANEK.PL

HTTP://BLOG.ADAMFURMANEK.PL

FURMANEKADAM