# Blockchain Challenges: Energy Impact and Scalability

Prof. Dr. Hans P. Reiser,
Christian Berger

University of Passau

UNIVERSITÄT PASSAU

BBC | Sign in | Home | News | Sport | Reel | Worklife

NEWS

Home | Coronavirus | Video | World | UK | Business | Tech | Science | Stories | Entertai

Tech

# Bitcoin Mining Council to report renewable energy usage

1 day ago


**A new Bitcoin Mining Council has been creat currency's sustainability, following a meetin Elon Musk.**

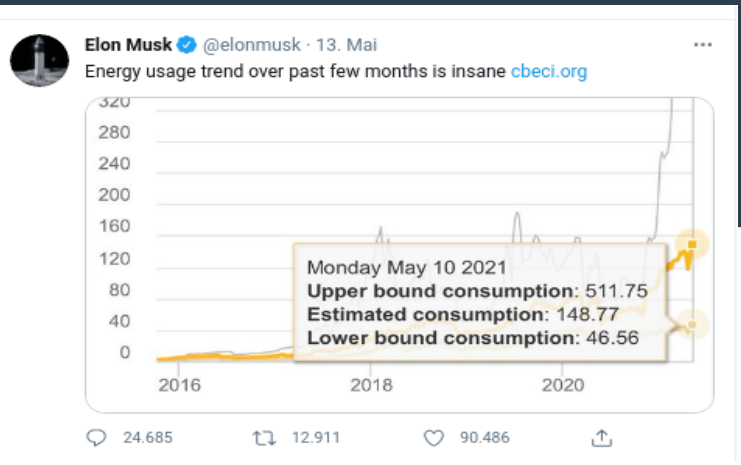
The Tesla CEO tweeted the development was

It's hoped the council will "promote energy u encourage miners to use renewable sources.

The process of creating Bitcoin consumes lar

Its value fell earlier this month after Tesla wit currency, **citing environmental concerns.**

News source:
https://www.bbc.com/news/




CNBC | Sea

MARKETS | BUSINESS | INVESTING | TECH | POLITICS | CNBC TV | WATCHLIST | P

TECH

# Elon Musk says Tesla will stop accepting bitcoin for car purchases, citing environmental concerns

PUBLISHED WED, MAY 12 2021·6:20 PM EDT | UPDATED WED, MAY 12 2021·8:26 PM EDT

Lora Kolodny
@LORAKOLODNY

**KEY POINTS**

- Tesla has "suspended vehicle purchases using bitcoin," out of concern over "rapidly increasing use of fossil fuels for bitcoin mining," according to a tweet from CEO Elon Musk on Wednesday.

News source:
https://www.cnbc.com/2021/05/12/elon-musk-says-tesla-will-stop-accepting-bitcoin-for-car-purchases.html

Policy

# China Tightens Crypto Mining Crackdown, Bans Trading

The country's top regulators are not done with crypto.

By Eliza Gkritsi · Sep 24, 2021 at 11:33 a.m. · Updated Sep 24, 2021 at 5:55 p.m.

# Could Ethereum Run On Just 0.05% Of It's Current Electricity Levels?! A Potential GAME CHANGER....

Silicon Valley Newsroom 3:30 PM No Comments

Ethereum developer Carl Beekhuizer says *'Ethereum's power-hungry days are numbered'* and explores the energy usage difference that will be seen when Ethereum makes the switch to proof-of-stake(PoS). This replaces the current network's validation method known as proof-of-work (PoW, aka traditional 'mining') and will allow the platform to run on just 0.05% of it's current power usage level, claims Beekuizer.

According to Digiconomist miners currently consume 44.49 TWh per year, and Beekuizer says that could go as low as 0.02 THw by changing to PoS.

News source:
https://www.globalcryptopress.com/2021/05/could-ethereum-run-off-just-005-of-its.html

NEWS PLAN YOUR VACCINE COVID-19 POLITICS U.S. NEWS WATCH NOW

TECH NEWS

# Cryptocurrency goes green: Could 'proof of stake' offer a solution to energy concerns?

Bitcoin relies on many computers to crunch difficult math problems. But it doesn't have to.

Mining rigs mine the Ethereum and Zilliqa cryptocurrencies at the Evobits crypto farm in Cluj-Napoca, Romania, on Jan. 22, 2021. Akos Stiller / Bloomberg via Getty Images

May 25, 2021, 6:56 PM CEST / Updated May 25, 2021, 6:58 PM CEST

**By Ezra Kaplan**

At any particular moment, thousands of computers around the world are humming away, crunching complex math problems that create and sustain bitcoin.

This network gives bitcoin its appeal: decentralized, always on and easily tradeable. But it also means the network is constantly using energy – a sticking point for many of the cryptocurrency's skeptics and critics. And it's not just a bitcoin problem. Other cryptocurrencies and blockchains including Ethereum have similar challenges.

The debate about bitcoin's environmental impact was elevated earlier this month when Tesla CEO Elon Musk, once one of the most notable bitcoin boosters, said his company would no longer accept it for the purchase of vehicles. He cited the use of fossil fuels for bitcoin mining as a reason.

News source:
https://www.nbcnews.com/tech/tech-news/cryptocurrency-goes-green-proof-stake-offer-solution-energy-concerns-rcna1030

# Outline of this talk

- **Blockchain Basics**
- **Proof-of-Work “Nakamoto Consensus”**
  - How does it work
  - Energy Impact & Sustainability
  - Novel PoW variants with little energy consumption
- **Bzyantine Fault-Tolerant (BFT) Consensus**
  - The general idea, Proof-of-Stake Blockchains
  - Energy Consumption, Performance and Scalability
  - Selected examples, Algorand and Avalanche
- **Our Current Research: The AWARE Protocol**
  - The AWARE approach, integration in Hyperledger Fabric

# What is a Blockchain? (1)

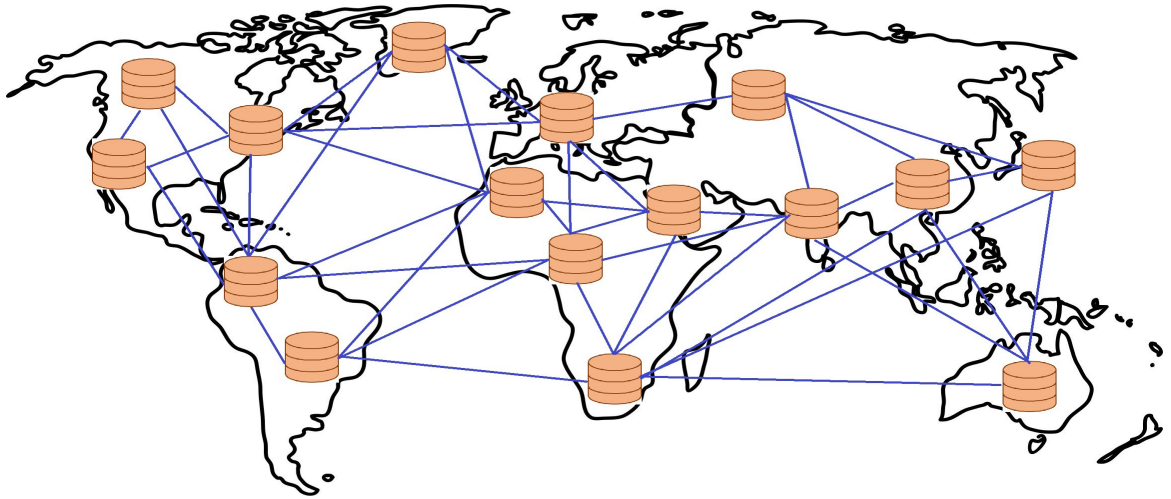
- **A distributed system that manages an**
  - append-only,
  - totally-ordered log
  - of immutable transactions (= the ledger)

  **in a “replicated fashion”**

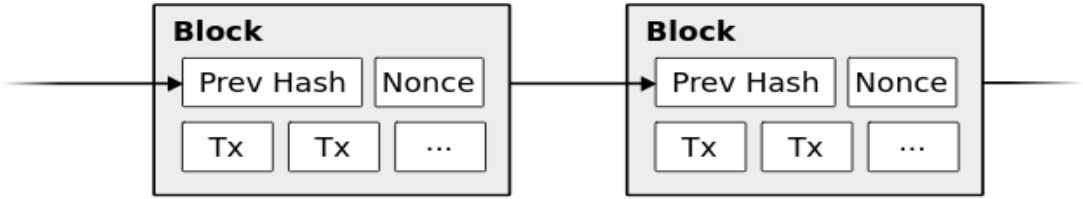
  what does that mean?

# What is a Blockchain? (2)

- **Several nodes**
  - hold a *consistent* copy of the ledger
  - are involved in *validating transactions*



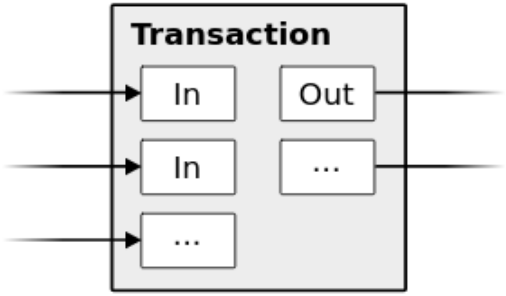
# What is a Blockchain? (3)




- **To order transactions,**
  - transactions are grouped into *blocks,*
  - blocks are *chained* by each block referencing the *hash* of the previous block,
  - a *consensus* primitive is employed to decide:

    ”which block should be appended next?”

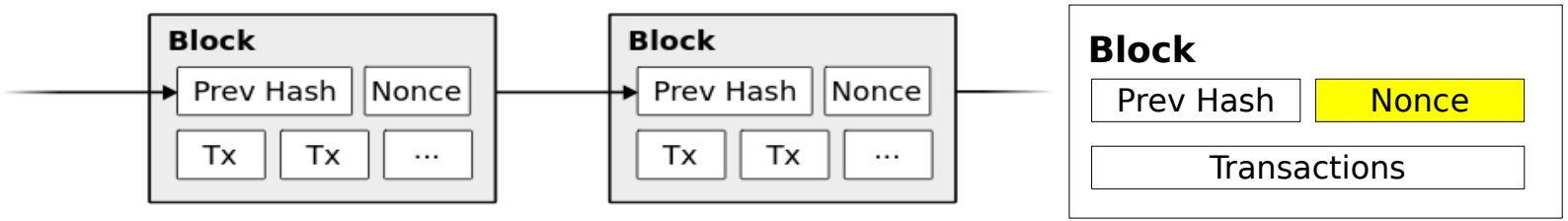
# Bitcoin Transaction Model




- **A *transaction* can combine or split value**
  - Inputs: Reference *Unspent Transaction Outputs* (UTXO)
  - Outputs: Creates new UTXOs
    - A receiver needs to be specified
    - Change can be returned to the sender in a second Output
    - Unspecified coins can be rewarded to the miner as *transaction fee*
      - Mining fee is optional but encourages a miner to prioritize a transaction

# Proof of Work: “Nakamoto Consensus”

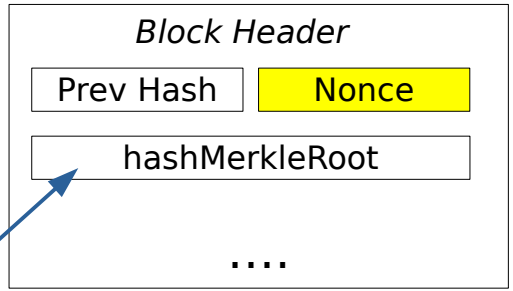
- **The Proof of Work**
  - Miner chooses transactions (but must be valid)
    - e.g., no conflicts, such as using the same UTXO as Input twice
  - Problem: finding a “fitting” Nonce
  - “Found” Blocks are broadcasted in the network
    - using Gossip
  - Invalid blocks are rejected!



# Proof of Work: Finding a Nonce

- **Finding a Nonce is hard**
  - We need to “try out” many nonces
  - We *hash* H() the block header and check:

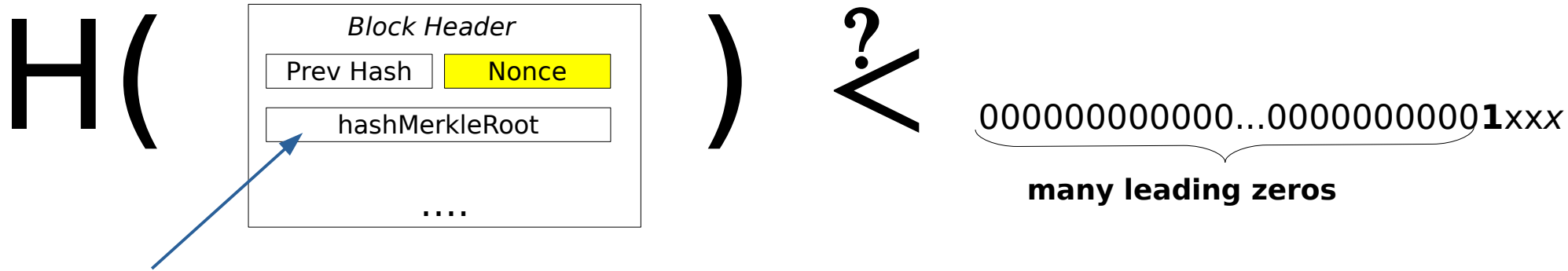
$$H(\ \text{Block Header}\ ) \stackrel{?}{<} thres$$




*hashMerkelRoot* is used instead of including *all transactions*

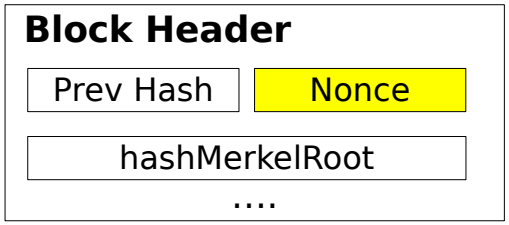
# Proof of Work: Finding a Nonce

- **Finding a Nonce is hard**
  - We need to “try out” many nonces
  - We *hash* H() the block header and check:




*hashMerkelRoot* is used instead of including *all transactions*

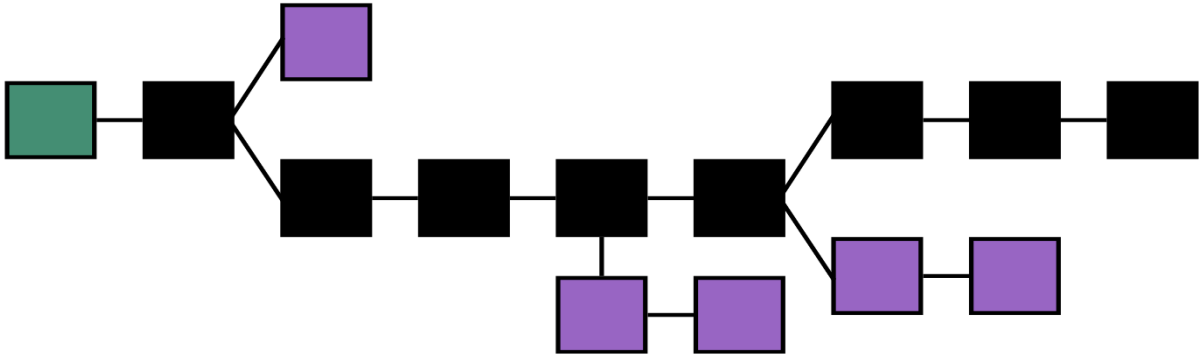
# Proof of Work: Mining

H( 

) $\stackrel{?}{<}$ *thres*

- **A large number of hashes needs to be calculated**
  - Because hash functions are one-way and can not be reversed easily
- **threshold depends on the difficulty**
  - Difficulty is regularly adjusted
    - so block generation time is *roughly* 10 minutes
    - If more computational power becomes available (because of more miners joining the network), then difficulty increases
- **Incentive: get Bitcoins for validating Txs**
  - Miner gets transaction fees + block reward

# Longest Chain Rule



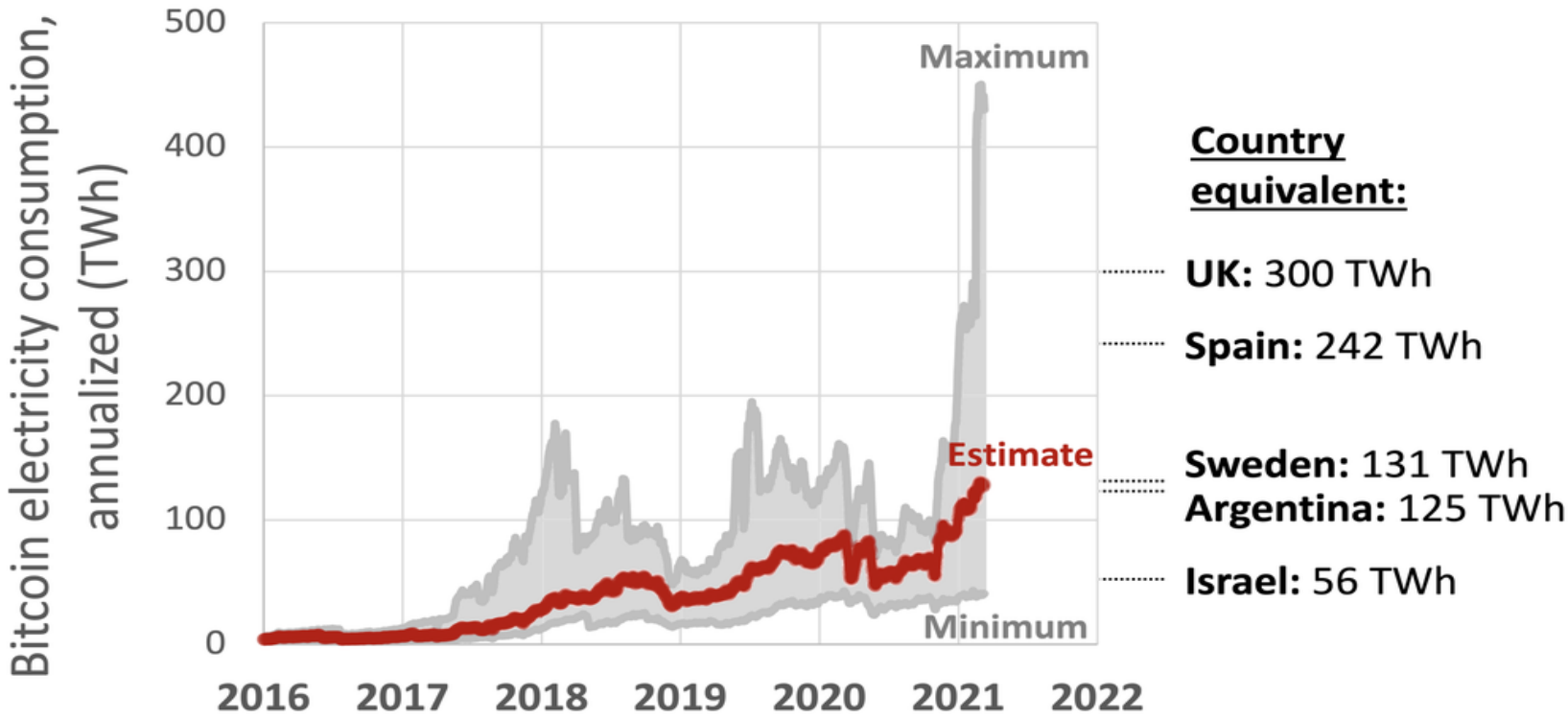
- **What happens if two miners find a solution at the same time?**
  - This can create forks of the Blockchain
  - Inbuilt conflict resolution mechanisms:
    - *“Longest chain always wins”*
    - Honest miners always switch to longest chain
      - Eventually majority will work on the same chain again as some fork will grow faster

# Bitcoin Mining

- **Bitcoin Mining is most efficient on ASICs**

- **Bitmain Antminer S19**
  - SHA-256
  - 95 Tera Hashes per second
  - Energy consumption **3250 Watt**
    - You need a cheap electricity price to become profitable
  - Costs only **9.250 €**

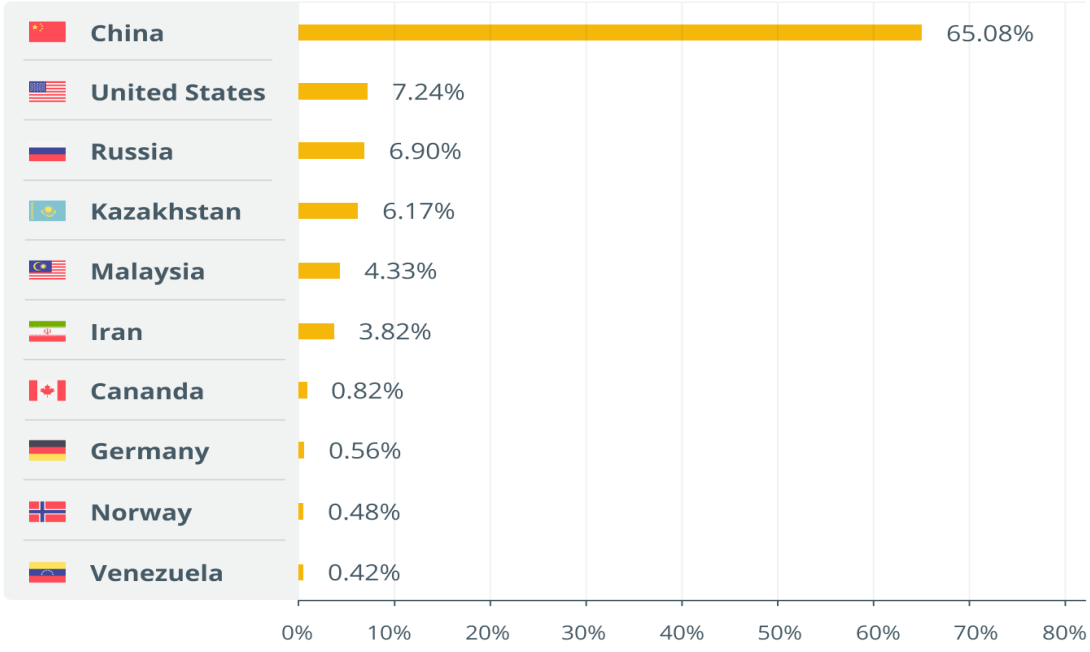
# Bitcoin: Energy Impact




Source: https://cbeci.org/

# Bitcoin Hashrate per Country

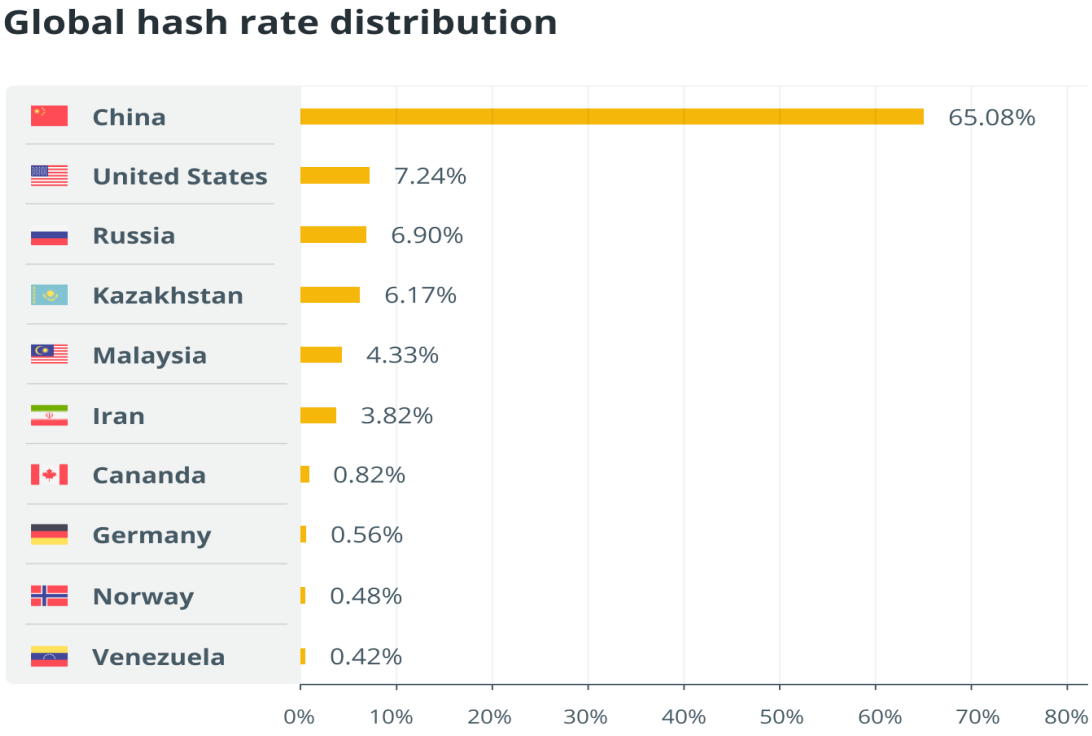
## Global hash rate distribution




cointelegraph.com

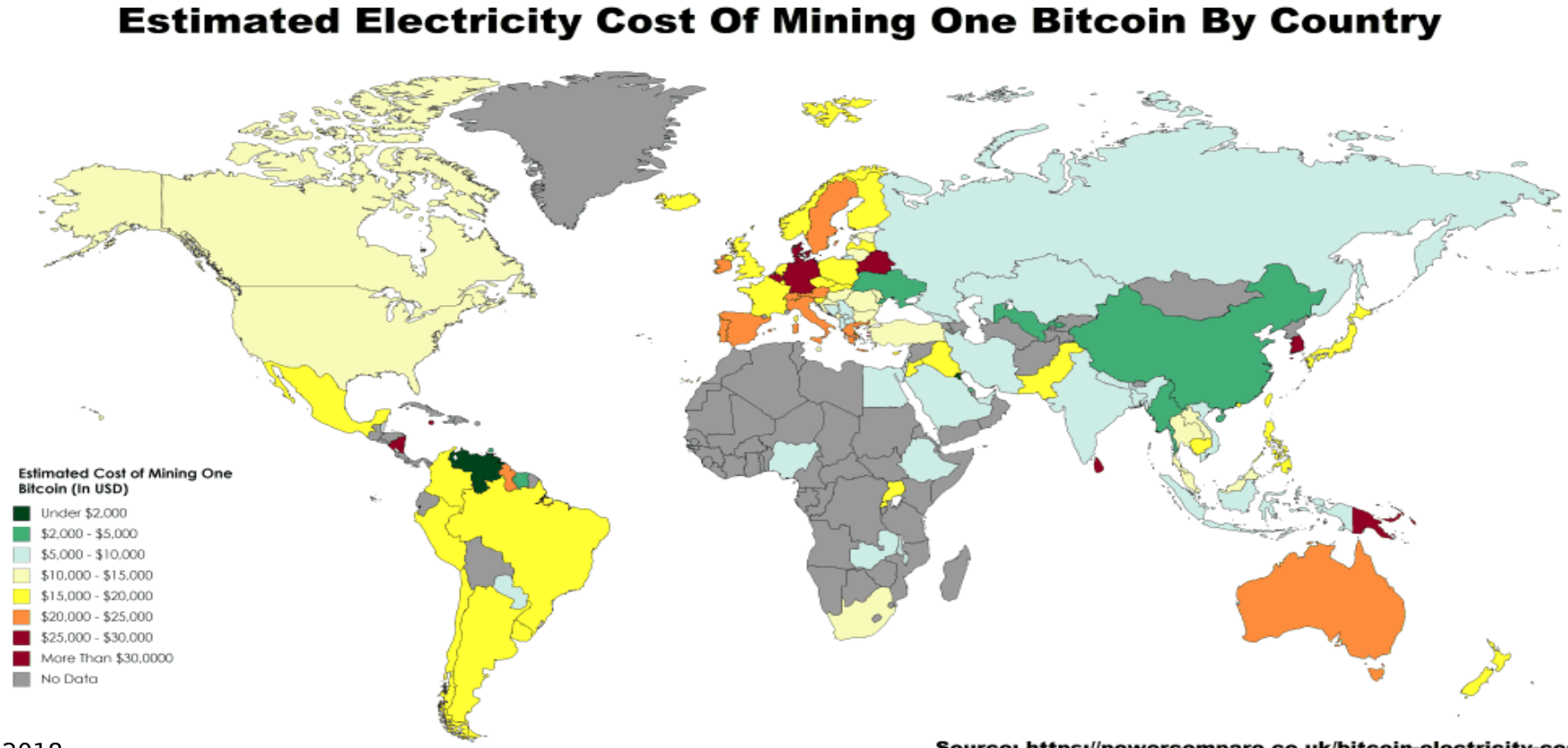
source: *Cbeci.org*

# Bitcoin Hashrate per Country




**Discussion: How can we explain this distribution?**

# Energy Costs per Country



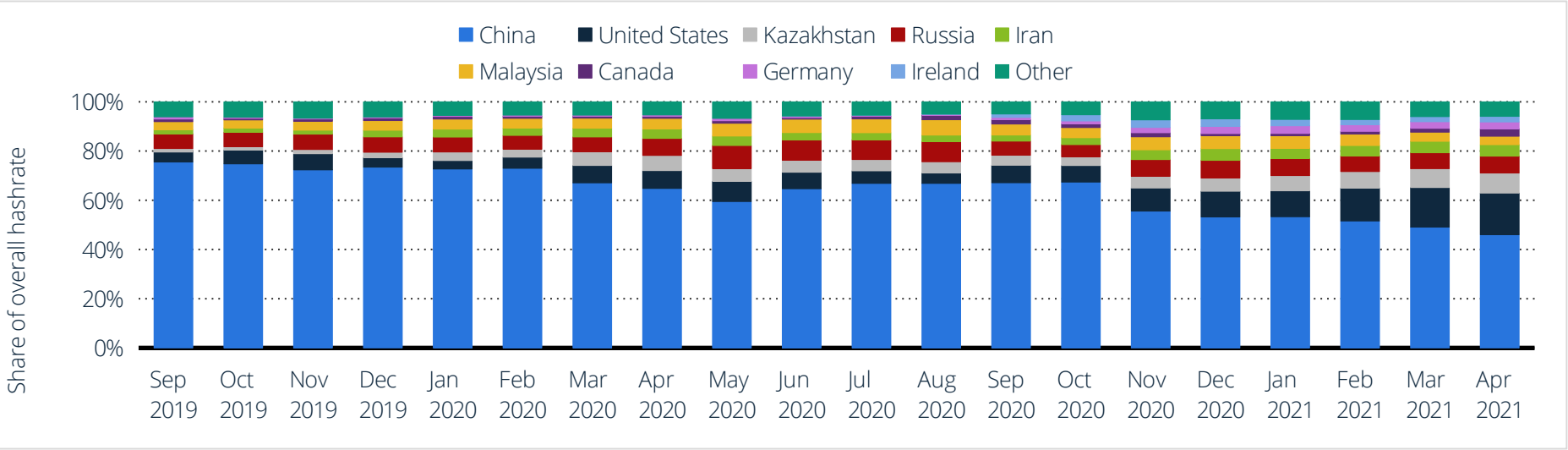

In March 2018

Source: https://powercompare.co.uk/bitcoin-electricity-cost/

# Bitcoin Hashrate per Country

## Distribution of Bitcoin mining hashrate from September 2019 to April 2021, by country

Countries that mine the most Bitcoin (BTC) 2019-2021

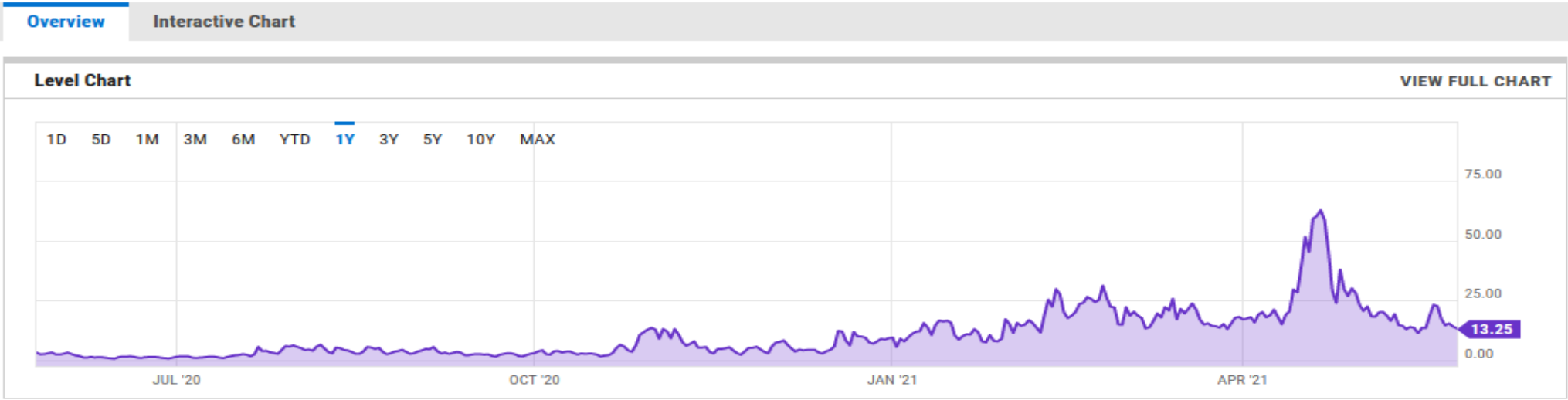


# Sustainability:
# Bitcoin Transaction CO2

- **A single bitcoin transaction has a carbon footprint of 734,69 kg CO2**
  - equivalent to the carbon footprint of 1.628.322 VISA transactions
  - Or 122.448 hours of watching YouTube
  - or driving a gasoline car ca. 3000 km

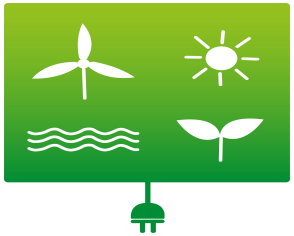
Estimated by Digiconomist
https://digiconomist.net/bitcoin-energy-consumption/

**Bitcoin Average Transaction Fee**
13.25 USD/tx for May 26 2021



# Can we solve the environmental problem of Bitcoin?

- **Use Green Energy instead of coal**
  - This is currently discussed: CO2-neutral PoW?
  - Yet, a huge energy waste still remains :(



- **Switch from Proof-of-Work to a different Consensus method**
  - e.g., Proof-of-Stake, Ethereum does this now!


- **Think about less energy-demanding Proof-of-Work variants or alternatives**
  - The “work” should not require energy, but other “resources” than computation, like time or disk space

chia

# Proof-of-Ellapsed-Time (PoET)

- **Work replaced**
  - with waiting for a random amount of time
- **Requires**
  - secure random number generation and attested proof of elapsed time
  - PoET program executed in trusted execution environment (Intel SGX)
- **Hyperledger Sawtooth**
  - currently provides support for PoET

# Proof-of-Space-and-Time (Chia)

- **Proof-of-Space**
  - A challenge is broadcasted in the network
  - Each “farmer” checks if they have the hash that is closest to the challenge
  - Probability of winning a a block is tied to space being available for farming
- **Proof-of-Time**
  - implemented by a Verifiable Delay Function
  - Sequential computation in which parallelisation does not yield benefits
  - A few such VDF servers are sufficient. Fastest honest server determines the speed
  - Eco-friendly as not much energy is needed

# Byzantine Fault-Tolerant (BFT) Consensus

- **Byzantine Faults**
  - Faulty components may behave arbitrarily
    - This includes “malicious behavior”

- **Consensus**
  - is a well-studied problem in distributed systems
  - Simply put:
    - *All correct* nodes need to eventually decide on a single, identical value v and make their decision only once
    - *v* must have been proposed by some node initially

# Byzantine Fault-Tolerant (BFT) Consensus

- **Byzantine Faults**
  - Faulty components may behave arbitrarily
    - This includes “malicious behavior”

- **Consensus**
  - is a well-studied problem in distributed systems
  - Simply put:
    - *All correct* nodes need to eventually decide (*Termination*) on a single, identical value *v* (*Agreement*) and make their decision only once (*Integrity*)
    - *v* must have been initially proposed (*Validity*) by some node

# Byzantine Fault-Tolerant (BFT) Consensus

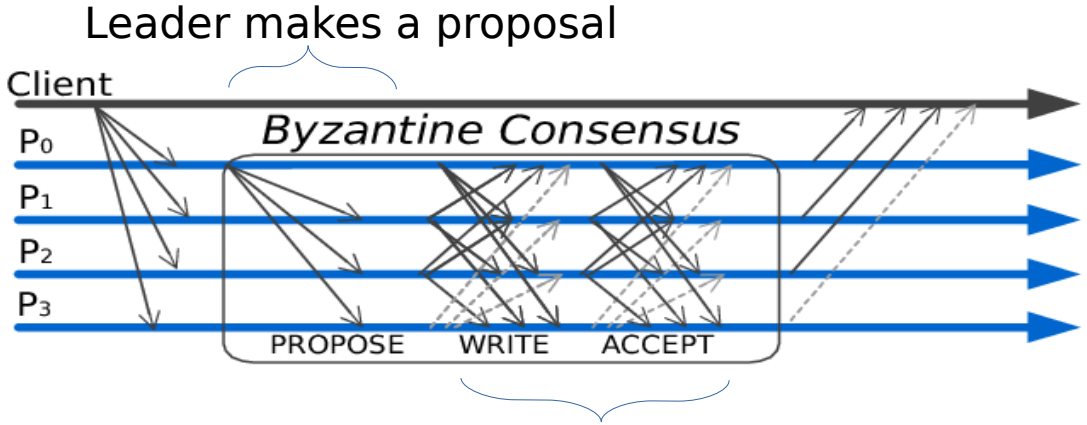
## Consensus

- is a well-studied problem in distributed systems
- Simply put:
  - *All correct* nodes need to eventually decide *(Termination)* on a single, identical value *v* *(Agreement)* and make their decision only once *(Integrity)*
  - *v* must have been initially proposed by some node *(Validity)*

**<u>Discussion:</u> Does Bitcoin's Proof-of-Work Consensus guarantee all these properties, too ?**

# Characteristics of BFT Consensus

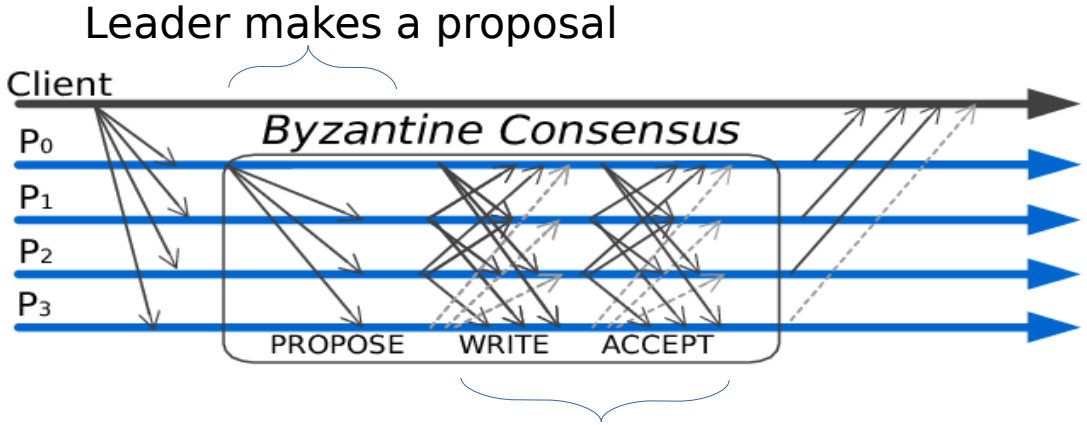
- **Communication-based:**
  - Nodes talk to each other to solve consensus




2-Phase commitment:
- Nodes collect votes
- Quorums ensure agreement

# Characteristics of BFT Consensus

- **Communication-based:**
  - Nodes talk to each other to solve consensus

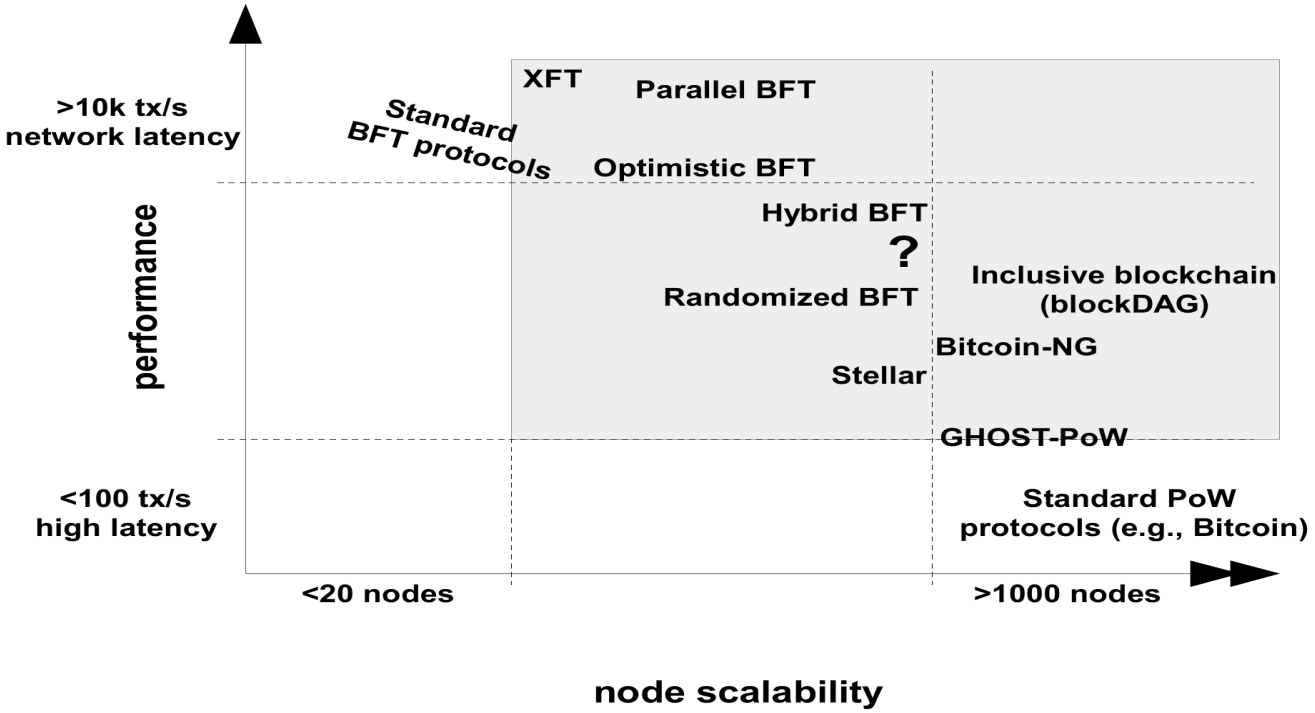



**<u>Discussion:</u> Why does this protocol scale worse in comparison to Bitcoin's Proof-of-Work ?**

# Characteristics of BFT Consensus

- **Communication-based:**
  - Nodes talk to each other to solve consensus
- **Energy-Efficiency**
  - Because consensus is not tied to a competition of wasting computation power
- **Performance is good**
  - High throughput (multiple thousand Tx/s)
  - Comparable low latency possible (a few seconds)

# BFT Replication vs. PoW




M. Vukolic: The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication

# BFT Replication vs. PoW

## Proof-of-Work

(+) scales well for a large number of nodes

(-) typically slow transaction speed and few throughput

(-) mining wastes energy

vs.

## BFT Consensus

(-) traditional protocols scale poorly for a large #nodes

(+) typically fast transaction speed and high throughput

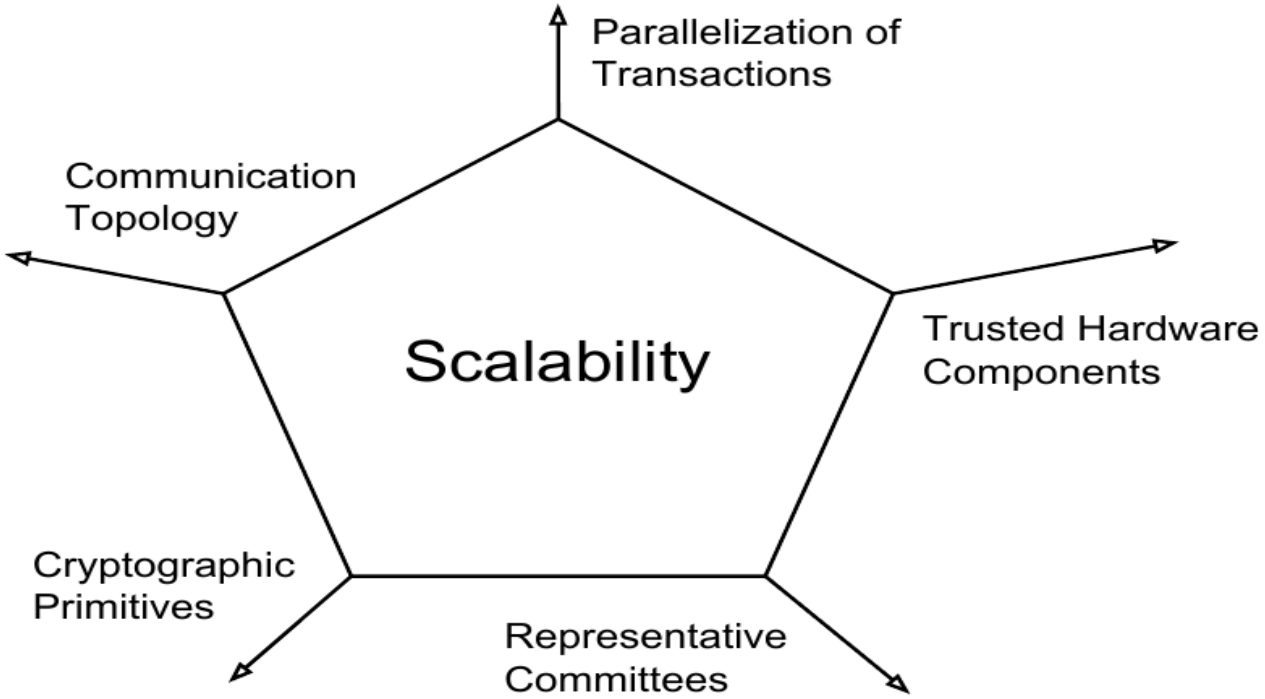
(+) works energy-efficiently

# BFT Consensus + Blockchain ?

- **But how can we maintain security in a permissionless environment ?**
  - Sybil Attack: An attacker may create a large number of fake identities in the network
  - Proof-of-Stake: Participation is coupled to having stake (native crypto currency of a blockchain)

- **Scalability**
  - Requiring a lot of communication / coordination between nodes limits the scalability of consensus
  - There are techniques for increasing scalability

# Scaling Byzantine Consenus



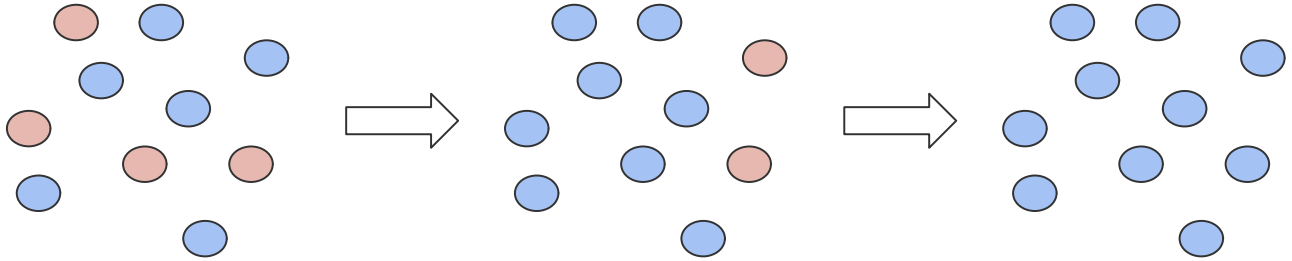
# Communication Topology

- Who talks with whom?
- **Key ideas**
  - flat-structured communication (HotStuff)
  - tree-structured communication (ByzCoin, Kauri)
  - overlay networks and gossip (Algorand, Gosig)
  - leader-less communication (Avalanche)
  - federated Byzantine agreement (Stellar)

# Leader-less Communication (Avalanche)

Avalanche's* idea relies in a metastable mechanism

- nodes repeatedly sample *k* randomly chosen other nodes and adapt their value to a certain majority
- thus, correct nodes are being guided towards the same consensus value



*Team Rocket. "Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies." (2018)

# Representative Committee

## Key idea:

- announce a committee of delegates with *active* roles (e.g. proposers and acceptors)

- a major portion of the nodes stay passive e.g. they only learn about the agreement value

- Important: the selection process should **not require coordination** among the nodes

# Representative Committee

## Cryptographic Sortition Algorithm*

- choosing a random subset of users according to per-user weights
- in a system with *n* users, for user *i* with weight $w_i$ the probability being chosen is proportional to

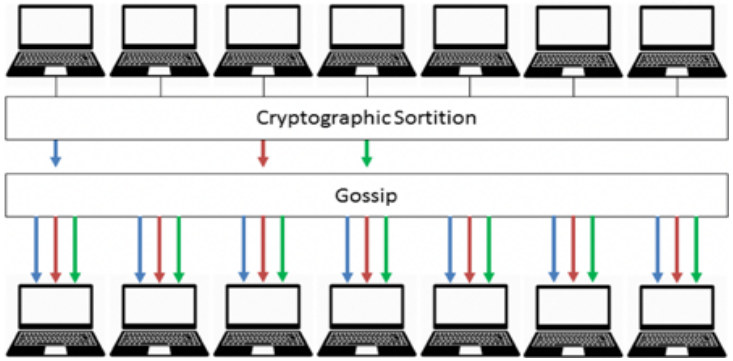
$$\frac{w_i}{\sum_{j=0}^{n-1} w_j}$$

- delegates are **chosen privately** to avoid being the target of an attacker

* Gilad, Yossi, et al. "Algorand: Scaling Byzantine agreements for cryptocurrencies." *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017.

# Algorand

- **Proof-of-Stake**
  - Gives reasonable security guarantees
- **Combines**
  - Committee (selected by Cryptographic Sortition)
  - Gossip



# Scalable Byzantine Consensus Protocols

- **are specifically designed for large-scale blockchain infrastructures**
- **differ in their assumptions and ambitions**
- **use and combine several novel techniques for scaling Byzantine consensus**

| | ByzCoin | FastBFT | Stellar | HoneyBadgerBFT | Algorand | Gosig | OmniLedger |
|---|---|---|---|---|---|---|---|
| **Scalability (evaluated with)** | 1004 | 199 | currently running ca 100 | 104 | **up to 500k** | up to 10k | 1800 |
| **Throughput (transactions/s)** | 700 (n=1004) | 370 (n=199) | 1000 (n ca. 100) | 1200 (n=104) | <1000 | 4000 (n=140) | **≥ 4000 (n=1800)** |
| **Latency** | 30s | **< 1s (1 Gbps LAN)** | few seconds | 100s | 1 minute | <1 minute | < 2s |
| **Synchrony** | weakly synchronous | weakly synchronous | asynchronous, but progress depends on synchrony | **asynchronous** | weakly synchronous | asynchronous, but provable liveness only under weak synchrony | synchronous |
| **Consensus determinism** | deterministic | deterministic | deterministic | probabilistic | probabilistic | probabilistic | probabilistic |
| **Approaches for scaling consensus** | communication tree + collective signatures | hardware-based TEE + secret sharing, tree topology | federal Byzantine agreement with hierarchical structure | novel ACS reduction with threshold encryption, efficient RBC with erasure codes | committee (cryptographic sortition) + gossip | multi-signatures + gossip | communication tree, collective signatures, parallelizing transactions |

Christian Berger, Hans P. Reiser. Scaling Byzantine Consensus: A Broad Analysis. The 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL) (co-located with the the 2018 ACM/IFIP/USENIX International Middleware Conference). December 2018.

# Permissioned Blockchains

- **Are based on a well-defined consortium of participators**
  - A “native” crypto currency or Proof-of-stake mechanism is not needed!
- **Employ “traditional” consensus mechanisms**
- **Build resilient distributed applications on top of it**
- **A popular open-source platform currently is Hyperledger Fabric**

# Conclusions

- **Proof-of-Work wastes a lot of energy**
  - This really can become an environmental problem
- **The energy problem of Blockchains can be solved**
  - Other novel alternatives based on time and/or space
  - Proof-of-Stake & Scalable BFT Consensus
  - Permissioned Blockchain & BFT Consensus

# Discussion

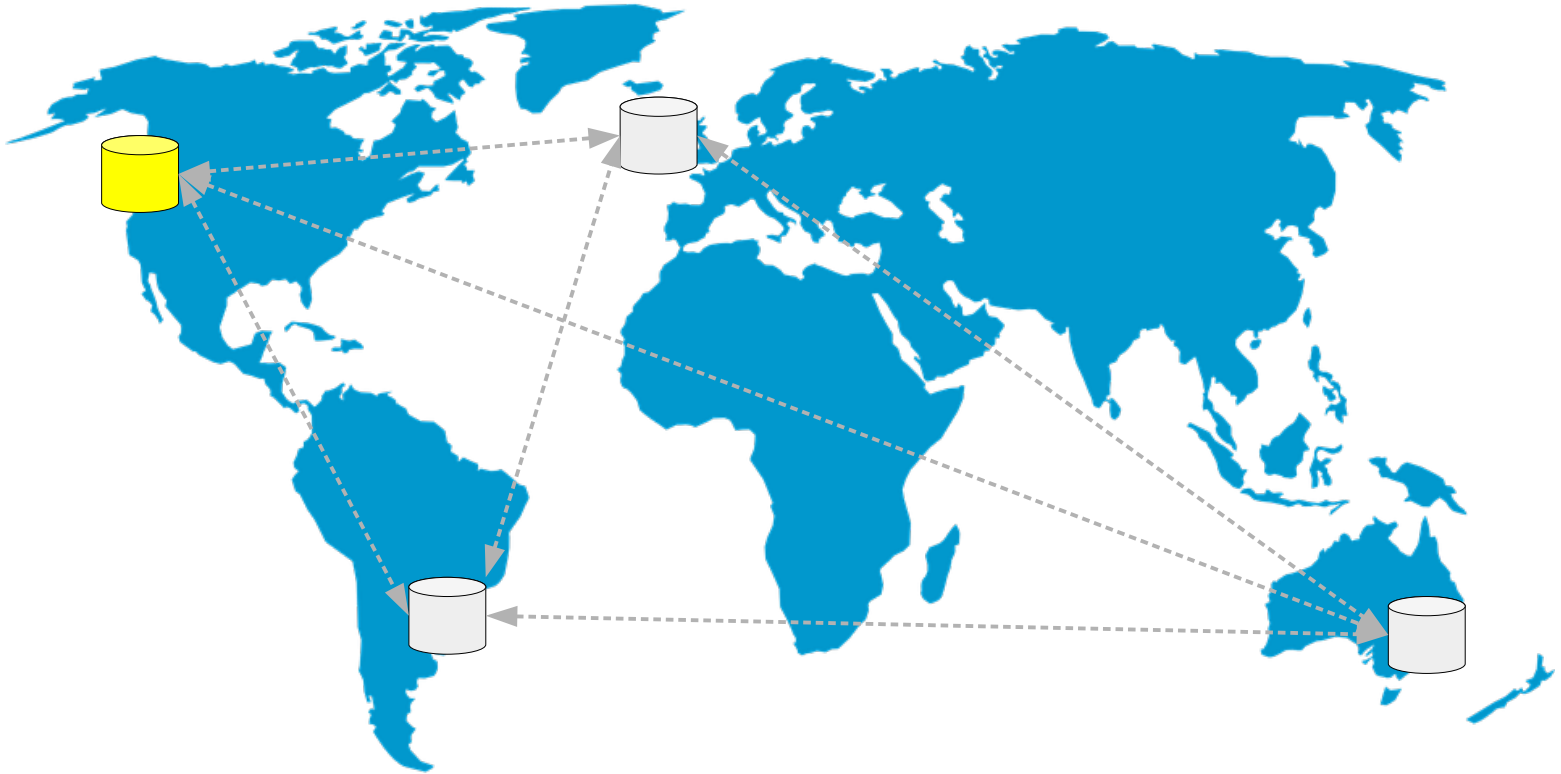
# Our Current Research

# Resilient Wide-Area Byzantine Consensus Using Adaptive Weighted Replication

Christian Berger*, Hans P. Reiser*, João Sousa**, Alysson Bessani**

*University of Passau, Germany
**LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

# Wide-Area Byzantine Consensus



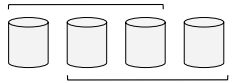
**System**

N=4, f=1

**Legend**

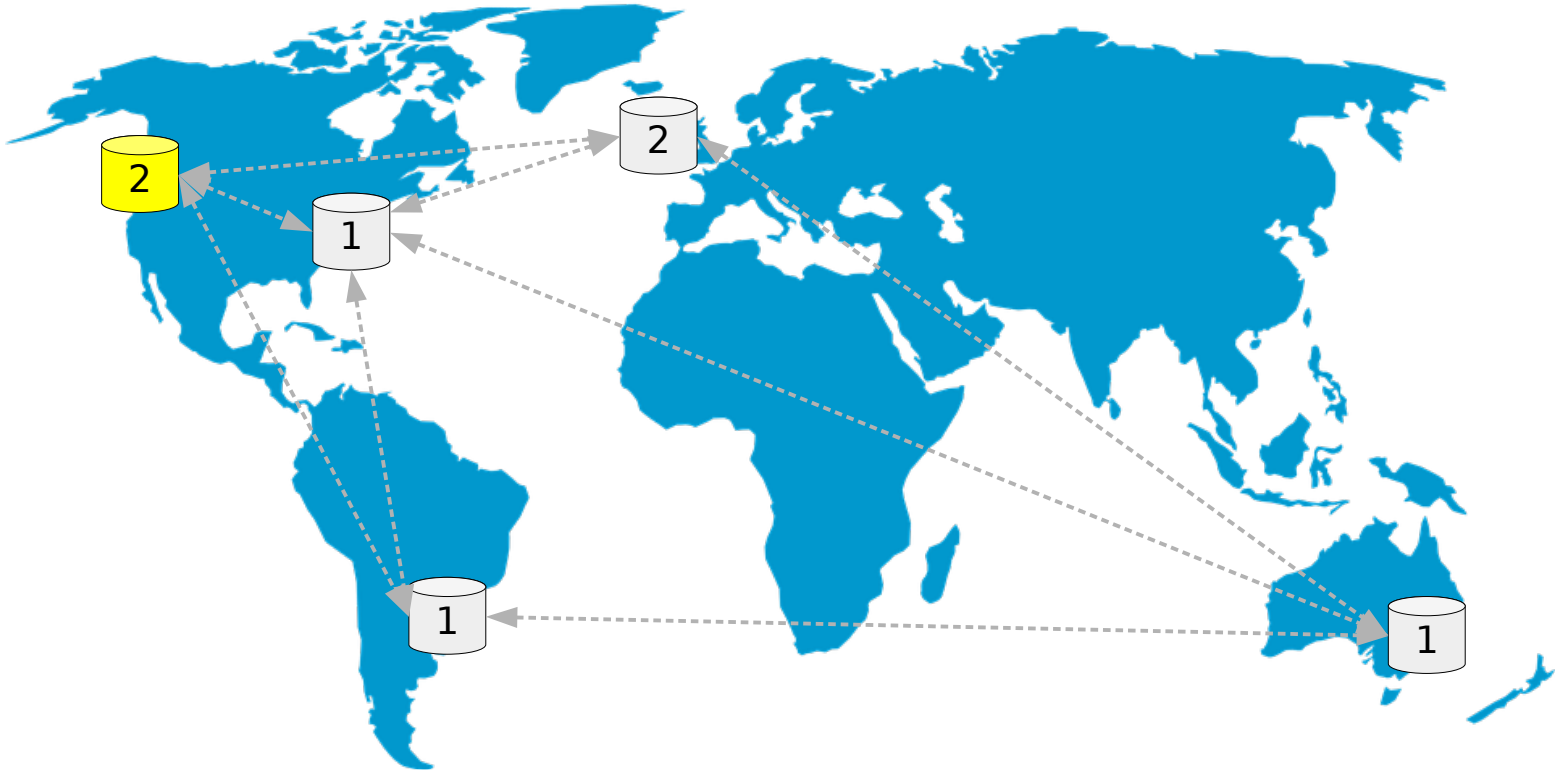
leader

**Quorum size**



Egalitarian quorums,
Any 3 out of 4 replica

# WHEAT: WeigHt-Enabled Active ReplicaTion*




**System**

N=5, **f=1,** Δ=1

**Legend**

leader

x — replica has voting power x

**Quorum size**

5 votes, 3 replicas

2 2 1 1 1

5 votes, 4 replicas

Weighted quorums

* Sousa, João, and Alysson Bessani. "Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines." *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015.

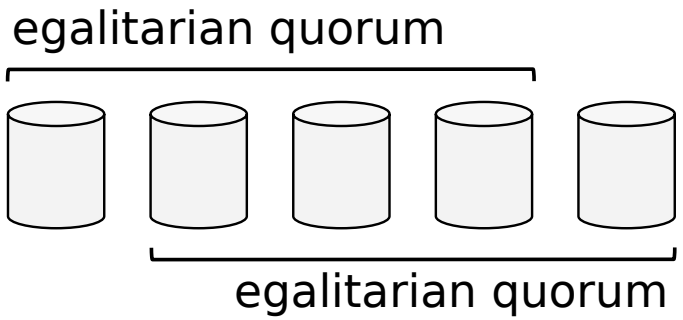
# Purpose of doing this? Latency gains!

- **To improve latency, we need to make enhancements on protocol level**
- WHEAT
  - utilizes the heterogeneous latencies of links between replicas
  - assigns higher voting power to well-connected replicas
  - benefits from more variety in quorum formation
  - allows replicas to faster make progress by accessing a proportionally smaller quorum of replicas

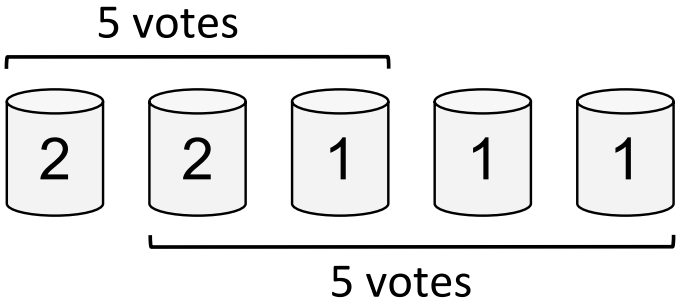
* Sousa, João, and Alysson Bessani. "Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines." *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015.

# Weighted Replication

- Weighted replication is **safe** and **does not violate the resilience bound f**
- Possible quorums for a n=5, f=1 system:



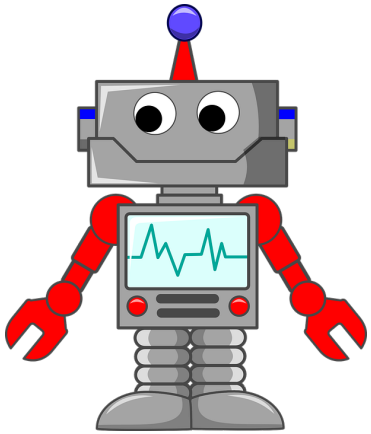

**Egalitarian:** all quorums contain $\lceil \frac{n+f+1}{2} \rceil$ replicas.




**Weighted:** a quorum contains at most *n - f* and at least *2f + 1* replicas.

# Remaining challenges? Automation needed!

- The benefit of weighted replication depends on **choosing an optimal weight configuration** (a non-trivial problem!)
- The environment of the system (i.e. network characteristics) may **change at runtime** (e.g., due to a DDoS attack)



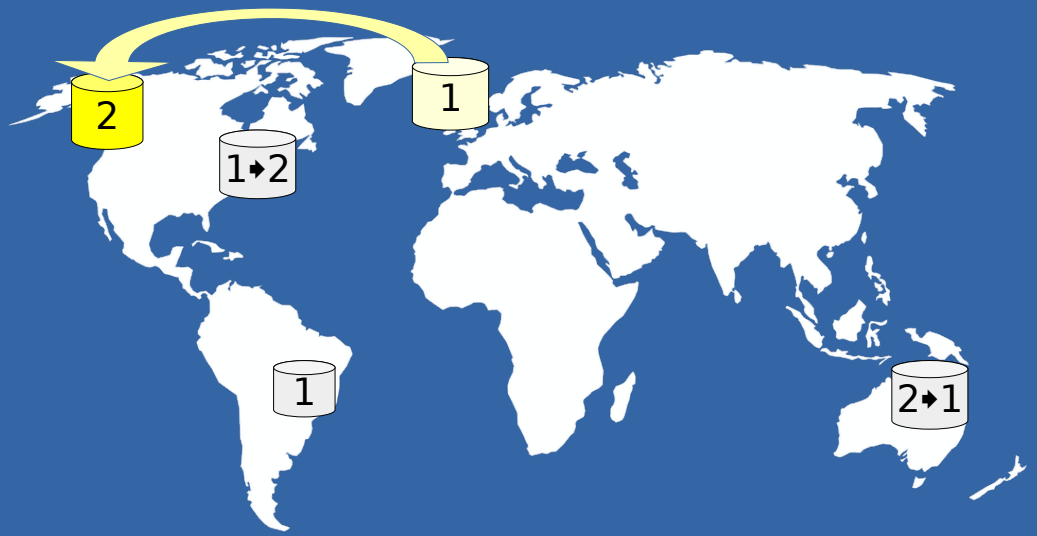
**AWARE** (**A**daptive **W**ide-**A**rea **RE**plication) enables a geo-replicated system to **adapt to its environment!**

# Practical Use?

- Recent blockchain developments (e.g., Libra, Tendermint, Hyperledger) might employ Byzantine consensus in a geographically distributed environment
  - Then they could benefit from **adaptive, weighted replication**

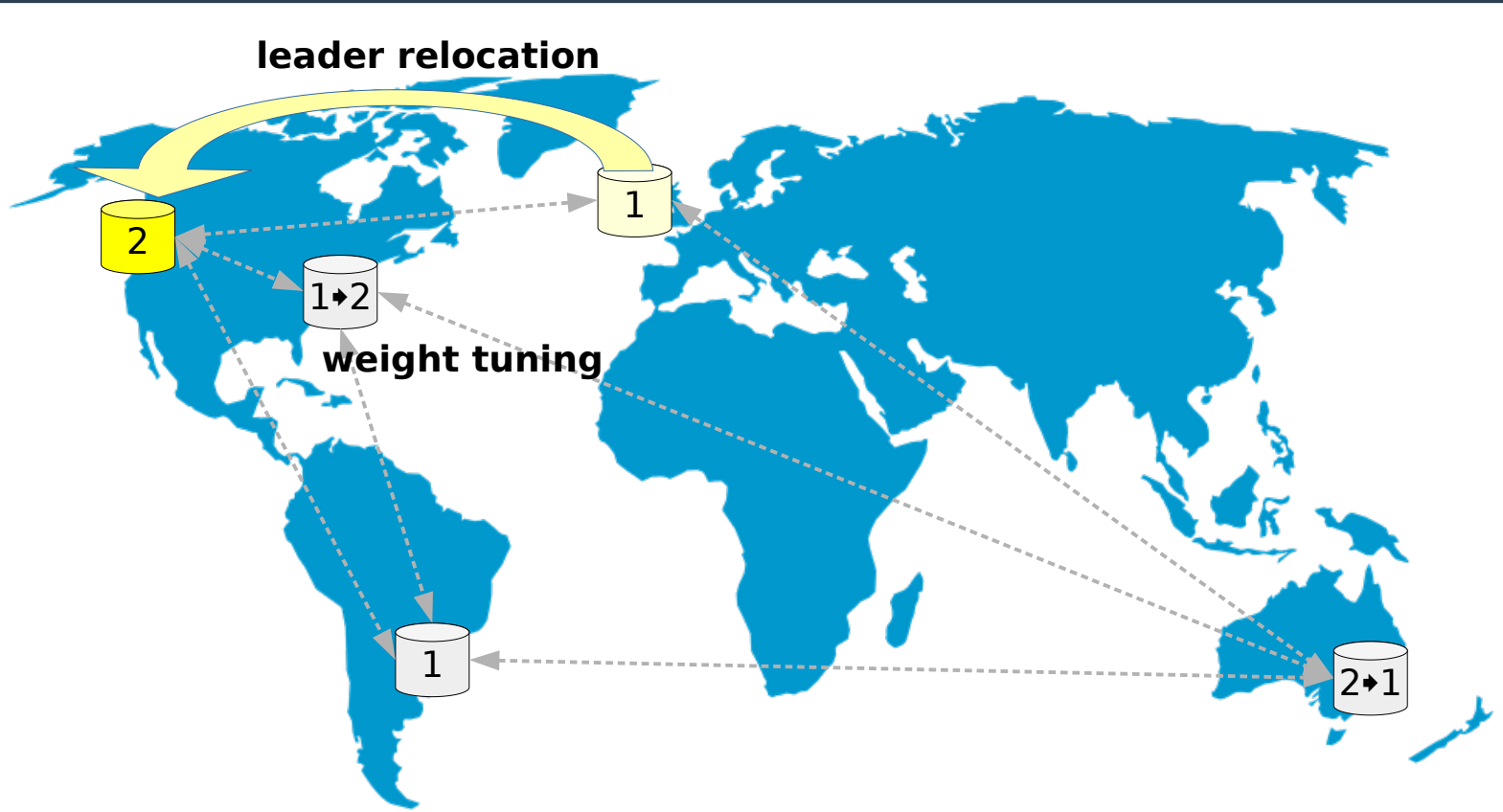
- AWARE can be used as basis for consortium-based blockchains






# The AWARE protocol

# Resilient Wide-Area Byzantine Consensus Using Adaptive Weighted Replication




## System

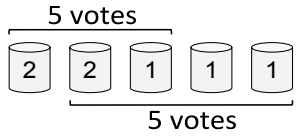
N=5, **f=1,** Δ=1

## Legend

leader

x replica has voting power x

## Quorum size




## Adapt to environment

1➧2 voting weight tuning

leader relocation

# AWARE Approach

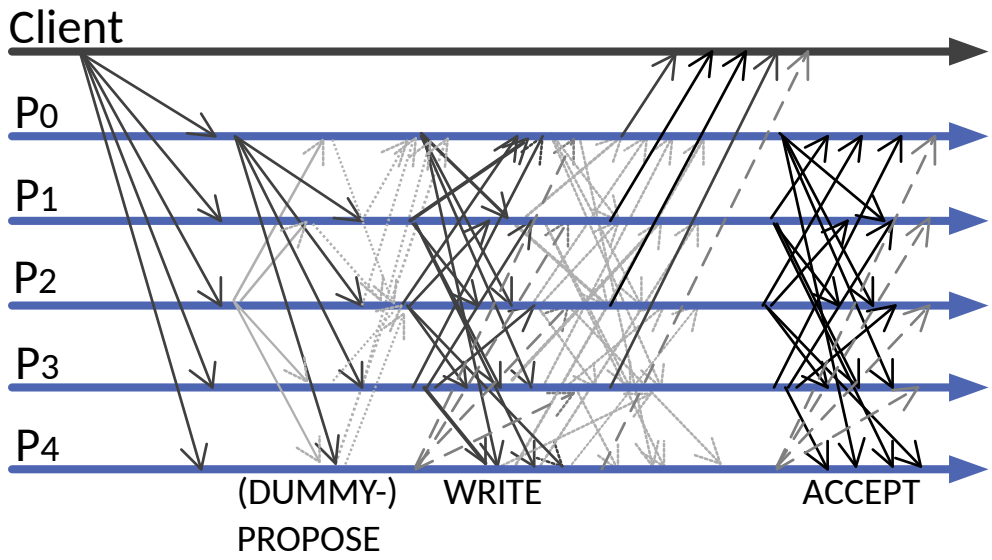

- **Monitoring**
  - AWARE uses reliable self-monitoring as decision-making basis for adapting replicas‘ voting weights and leader position at runtime
- **Self-Optimization**
  - AWARE continuously strives for latency gains at runtime
  - It optimizes voting weights and leader position to minimize consensus latency

# Self-Monitoring: Measuring Latency

- **Measuring latency**: Each replica measures its point-to-point latency to all other replicas from its own perspective for consensus protocol messages

- **Non-Leader's Propose**
  - Periodically an alternately selected dummy leader broadcasts a dummy proposal

- **Write-Response**
  - Replicas immediately respond by sending acknowledgments



# Self-Monitoring: Disseminating Measurements

- **Dissemination of measurements**
  - Replicas periodically disseminate their measurements with **total order**
  - Replicas maintain the same latency matrix after some specific consensus instance
  - AWARE maintains **synchronized matrices** for both Propose and Write latencies $\hat{M}^P$ and $\hat{M}^W$ used for decisions later

| | Oregon | Ireland | Sydney | Sao Paulo | Virginia |
|---|---|---|---|---|---|
| **Oregon** | 0 | 65 | 69 | 92 | 40 |
| **Ireland** | 65 | 0 | 132 | 93 | 38 |
| **Sydney** | 69 | 132 | 0 | 158 | 105 |
| **Sao Paulo** | 92 | 93 | 158 | 0 | 61 |
| **Virginia** | 40 | 38 | 105 | 61 | 0 |

# Self-Optimization

- Assume replicas have a synchronized, sanitized latency matrix $\hat{M}$

- When a defined number of consensus is reached, each replica **deterministically** solves the following optimization problem:

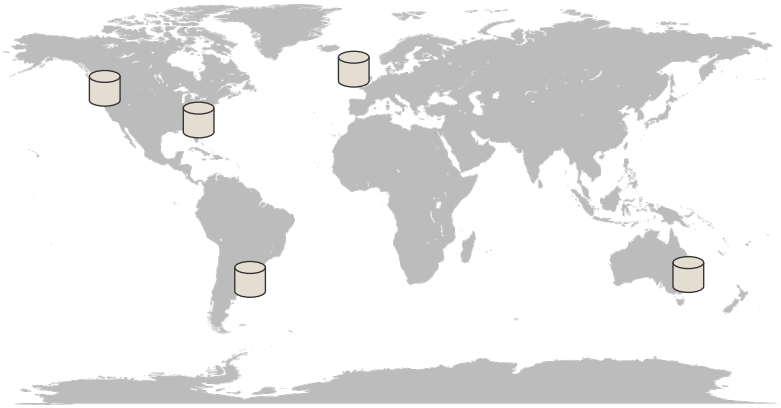
$$\langle \hat{l}, \hat{W} \rangle = \underset{W \in \mathfrak{W}, l \in \mathfrak{L}}{\arg\min} \; PredictLatency(l, W, \hat{M}^P, \hat{M}^W)$$

- All replicas reach the same, optimal weight distribution

# Evaluation

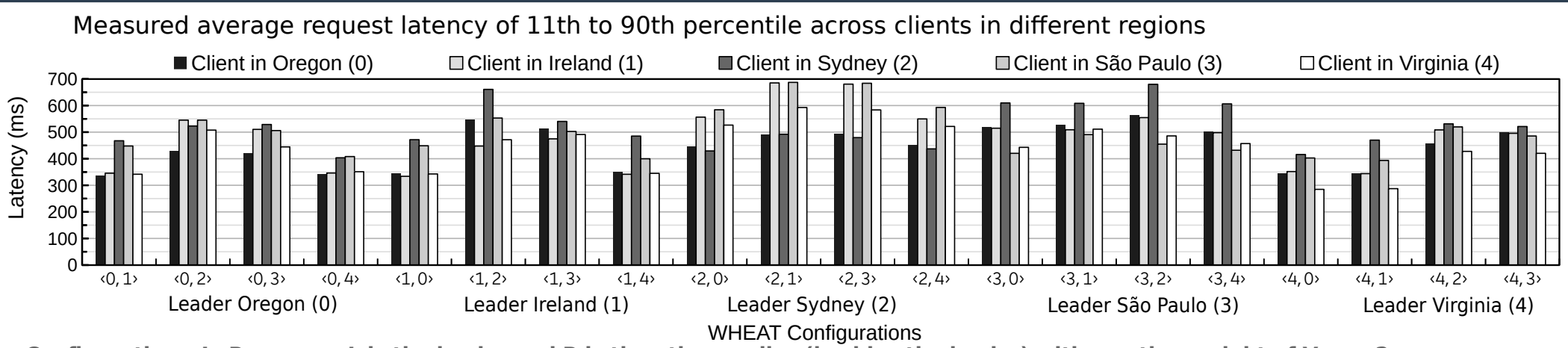
# Setup

- **AWARE*** is implemented on top of WHEAT, which is based on BFT-SMaRt
- For evaluation, we use the **Amazon AWS cloud**, using EC2 instances of t2.micro type with 1 vCPU, 1 GB RAM and 8 GB SSD volume
- We select regions **Oregon, Ireland, Sydney, São Paulo** and **Virginia** for instances (1 client and 1 replica on each instance)
- Clients simultaneously send requests across all sites



*Code of AWARE prototype is available at https://gitlab.sec.uni-passau.de/cb/aware

# Clients' Observed Request Latency



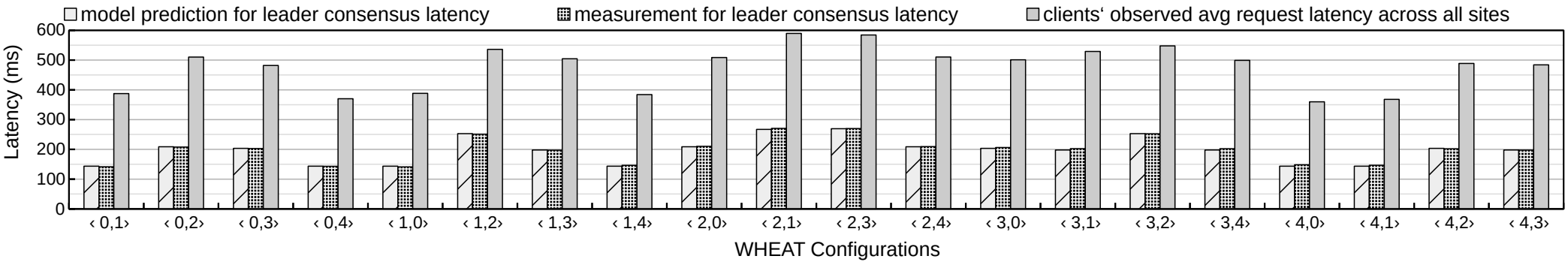

**Configuration <L, R> means L is the leader and R is the other replica (besides the leader) with a voting weight of $V_{max} = 2$**

## Observations

- The best configuration <4,0> performs about 38.7% faster than the median <3,4>, 63.9% faster than the worst <2,1>
- Tuning voting weights can reduce latency (compare configurations with the same leader)
- Leader relocation may be necessary for achieving optimal consensus latency

# Accuracy of Consensus Latency Prediction

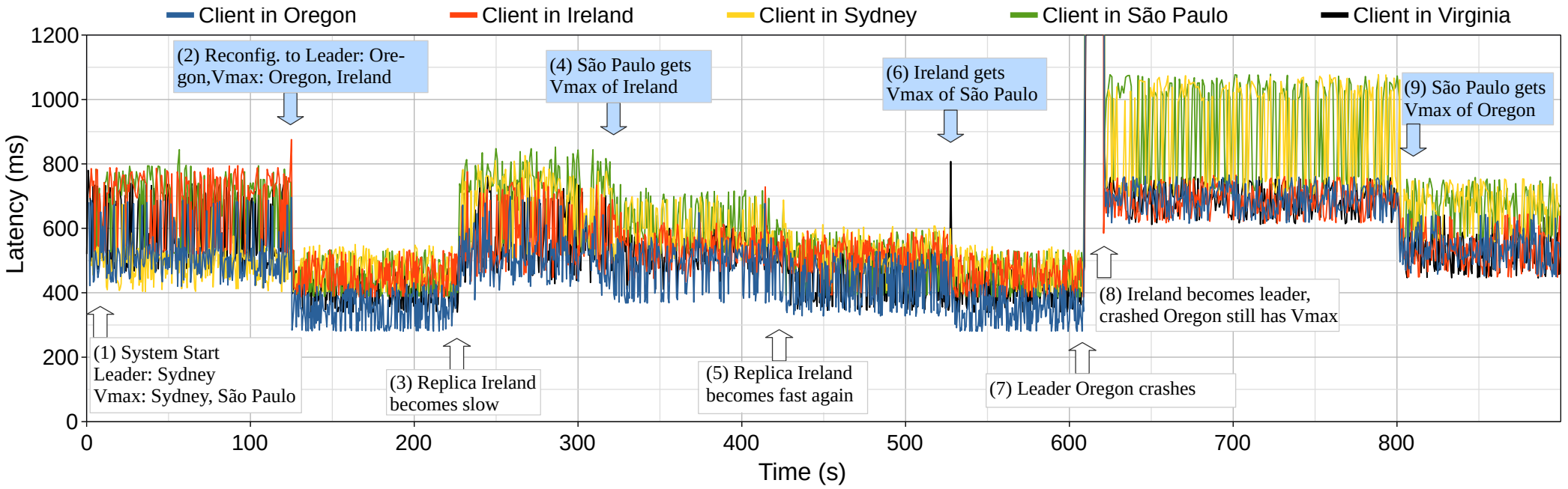
Comparison between predicted consensus latency, measured consensus latency and clients' observed average latency




## Observations

- Accuracy of model prediction with respect to observed consensus latency of the leader: Predictions were off by 1.08% on average
- Strong correlation between series of model latency predictions and clients' observed request latency, $\rho(L^{MP}, L^{CR}) = 0.961$

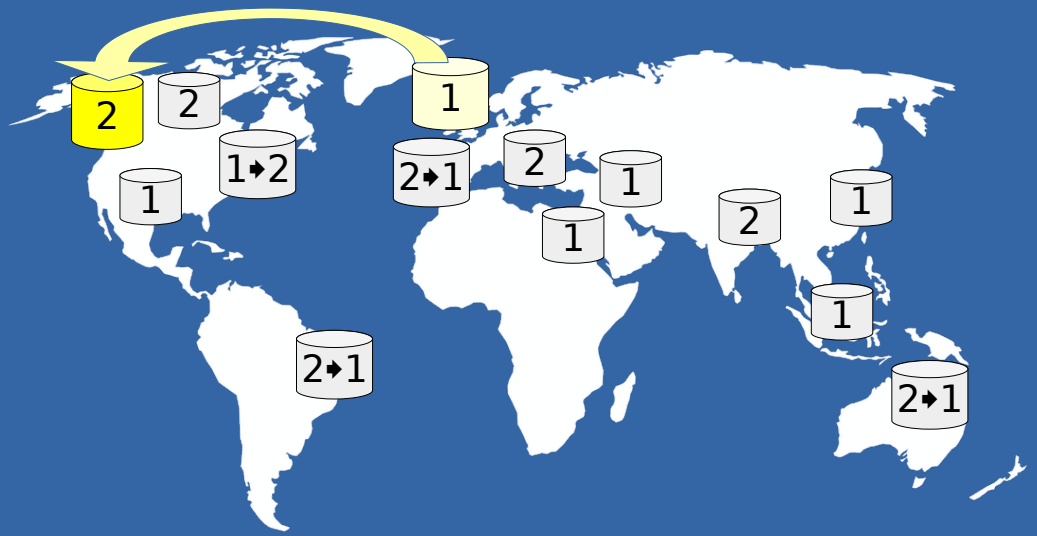
# Runtime behavior of AWARE




AWARE's automated reactions

# Summary of Observations

- **Ease of deployment**
  - AWARE provides the needed automation for finding an optimal configuration by tuning voting weights and/or relocating the leader
- **Adjusting to varying conditions**
  - AWARE dynamically adjusts to changing conditions by shifting high voting power to replicas that are the fastest in a recent time frame
- **Compensating for faults**
  - Up to $f$ replicas with high voting power become unavailable and hence restrict quorum variability
  - For non-malicious behavior, AWARE detects this and restores the availability of up to $f\ (V_{max} - V_{min})$ voting power by redistributing high voting weights

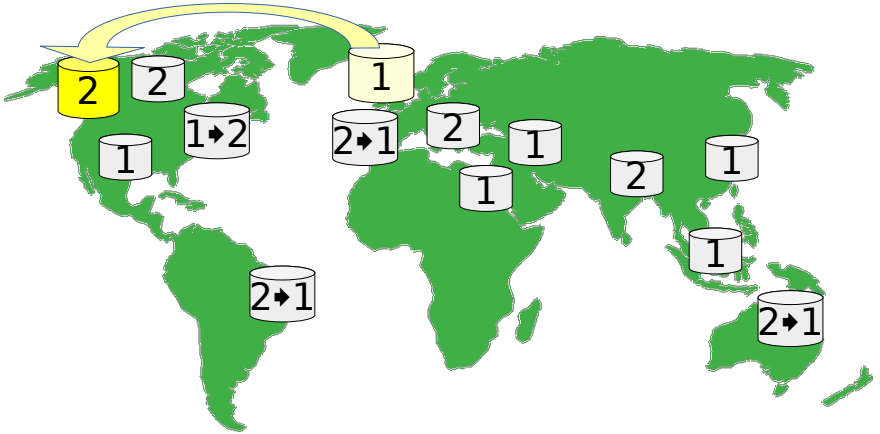



# AWARE in Larger Systems

# Motivation

- **Exploring the whole configuration space becomes increasingly difficult**
  - At some point it will become computationally infeasible to estimate all possible system configurations

- **Solution:** make the optimization method more scalable
  - We employ **Simulated Annealing** as heuristic
  - to approximate the best configuration

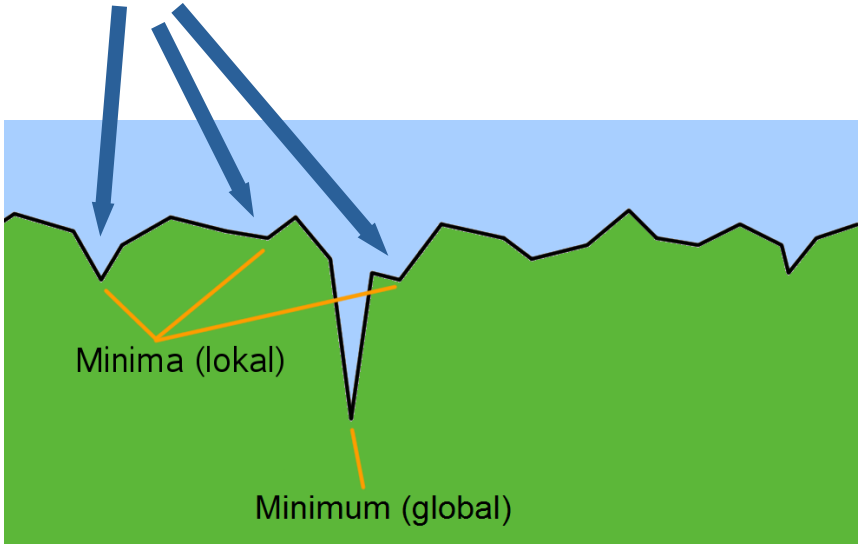


# Simulated Annealing (SA)

- **SA is essentially a local search**
  - that aims for improving on a current solution *c*
  - by randomly modifying it slightly (neighbor picking) to *c'*

- If *c'* is a better solution than *c*, proceed search with *c'*
- Else, compute acceptance probability to try out *c'* anyways
- Monotonically decreasing *temperature function temp*
- Exit condition: If *temp* is lower some *threshold*

**SA** can "jump" out of a local optimum with some probability to find the global optimum



# SA for AWARE Consensus Latency

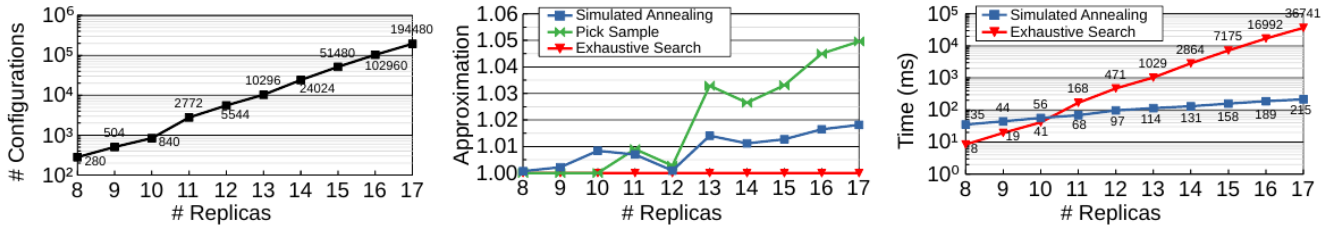
- **SA is essentially a local search**
  - that aims for improving on a current solution *c*
  - by randomly modifying it slightly (neighbor picking) to *c'*
- If *c'* is a better solution than *c*, proceed search with *c'*
- Else, compute acceptance probability to try out *c'* anyways
- Monotonically decreasing *temperature function temp*
- Exit condition: If *temp* is lower some *threshold*

**Algorithm 3:** *SimulatedAnnealing* is a heuristic for efficiently traversing the search space of configs $C$

**Data:** replica set $I$, system sizes $n$, $f$, $u$, $\Delta$, latency matrices for *PROPOSE* $\hat{M}^P$ and *WRITE* $\hat{M}^W$, consensus id *cid*, start temperature $t_0$, cooling rate $\theta$, temperature *threshold*
**Result:** best (approx.) performing configuration found

1 $c \leftarrow$ some $c_0 \in C$
2 $c.prediction \leftarrow predictLatency(I, c, \hat{M}^P, \hat{M}^W, n, f, \Delta)$;
3 $c_{approx} \leftarrow c$
4 $temp \leftarrow t_0$
5 $random \leftarrow$ new Random(cid)
6 **while** $temp > threshold$ **do**
7 /* Assign a $V_{max}$ to another replica */
8 $replicaFrom \leftarrow c.R_{max}[random.nextInt(u)]$
9 $replicaTo \leftarrow c.R_{min}[random.nextInt(n-u)]$
10 $c' \leftarrow c.swap(replicaFrom, replicaTo)$
11 **if** *replicaFrom is leader* **then**
12 $c'.setLeader(replicaTo)$
13 $c'.prediction \leftarrow predictLatency(I, c', \hat{M}^P, \hat{M}^W, n, f, \Delta)$;
14 /* If new solution is better, accept it */
15 **if** $c'.prediction < c.prediction$ **then**
16 $c \leftarrow c'$
17 **else**
18 /* Compute an acceptance probability */
19 $rand \leftarrow random.nextDouble()$
20 **if** $exp(\frac{-(c'.prediction - c.prediction)}{temp}) > rand$ **then**
21 $c \leftarrow c'$
22 **if** $c'.prediction < c_{approx}.prediction$ **then**
23 $c_{approx} \leftarrow c'$
24 /* Cool down the system */
25 $temp \leftarrow temp \cdot (1 - \theta)$
26 **return** $c_{approx}$

# Some Results

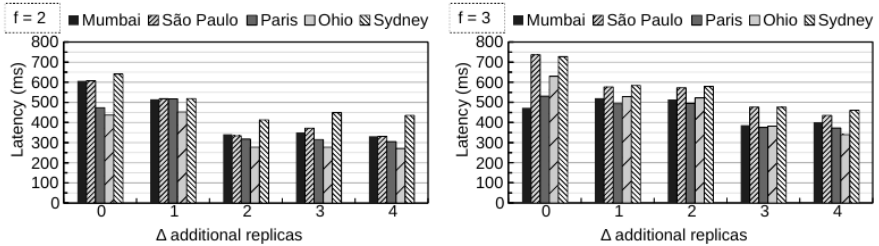



(a) # Configurations. (b) Approximation quality. (c) Computation time.

Figure 13: A heuristic can help to efficiently traverse the configuration space to find a good solution.

| | initial replica set | Δ-replicas |
|---|---|---|
| $f = 2$ | Mumbai, São Paulo, Paris, Ohio, Sydney, Tokio, Canada | Virginia, Ireland, Stockholm, Frankfurt |
| $f = 3$ | Mumbai, São Paulo, Paris, Ohio, Sydney, Tokio, Canada, Bahrain, Stockholm, Seoul | Frankfurt, Virginia, Ireland, Oregon |

(a) Test environments.



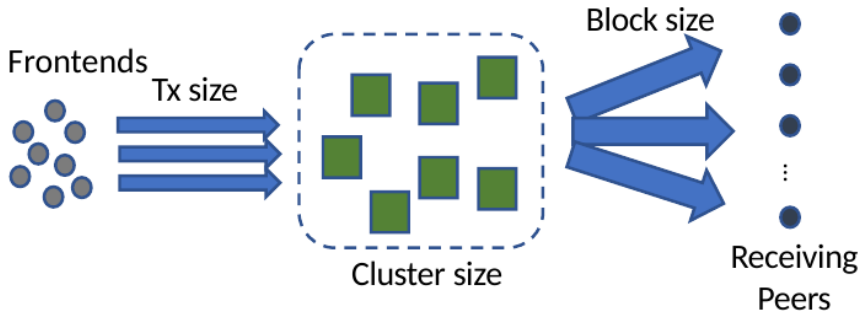

(b) $f = 2$ configuration. (c) $f = 3$ configuration.

Figure 14: Latency results of AWARE experimentally tested in larger-scale setups.

# AWARE in Hyperledger Fabric (HLF)

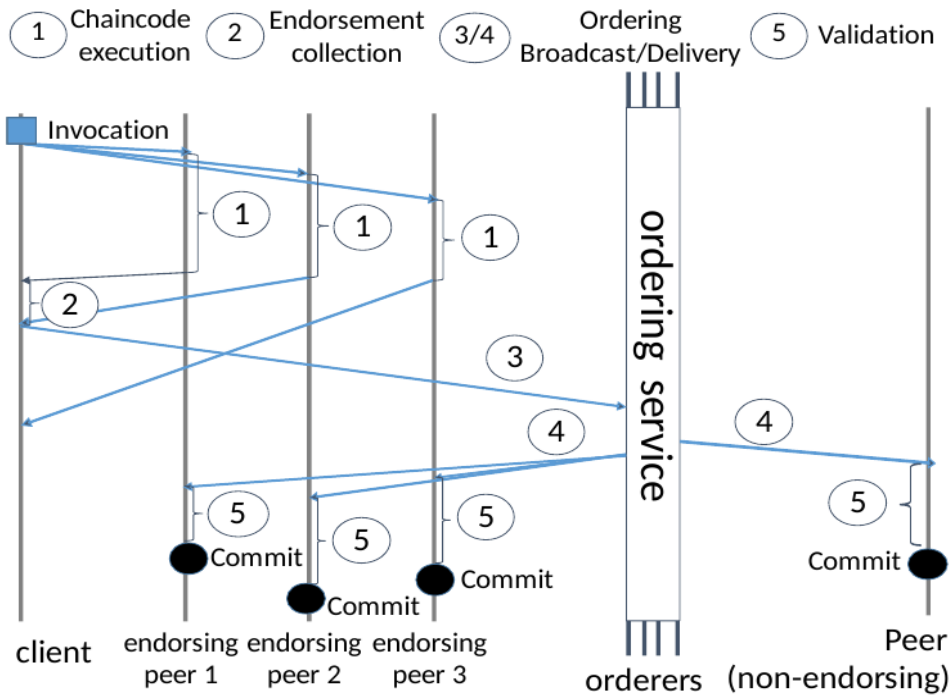
- We want to evaluate the AWARE ordering service for HLF
- Frontends located in different AWS regions:
  - Sydney (leader, V=2),
  - São Paulo (V=2),
  - California (V=1),
  - Tokio (V=1)
  - Stockholm (V=1)

Measure time needed for odering:




**(a) Ordering service performance model [20].**

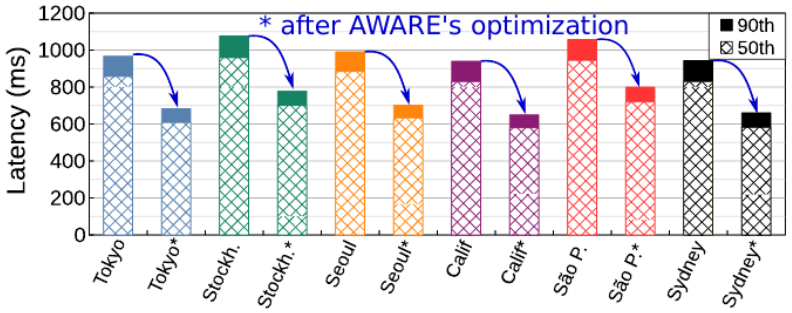
# Hyperledger Fabric



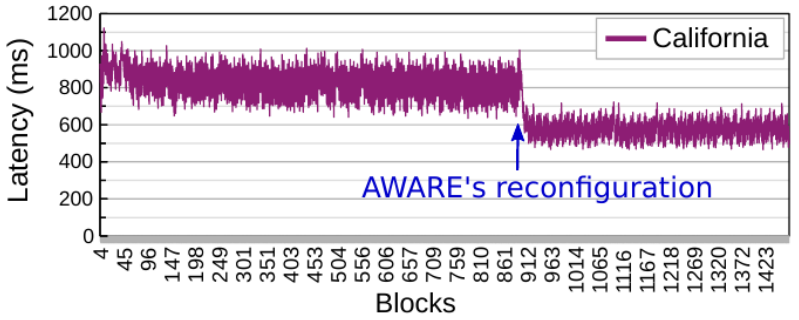

Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proc. of the 13th EuroSys Conf. pp. 1–15. ACM (2018)

# Results

- **AWARE reconfigures the system to optimize consensus latency**




(b) Latency gains across frontends that are deployed in different regions [5].




(c) Run-time observation of a frontend deployed in California [5].

Christian Berger, Hans P. Reiser, João Sousa, and Alysson Bessani.
AWARE: Adaptive Wide-Area Replication for Fast and Resilient Byzantine Consensus. IEEE Transactions on Dependable and Secure Computing (TDSC). Accepted in October 2020.

# Conclusions

# Conclusions

- World-spanning Byzantine consensus is getting practical and necessary with recent blockchain developments (e.g., Libra, Tendermint)

- AWARE enriches the idea of weighted replication
  - It provides the needed automation to adapt to changing environmental conditions → **adaptive weighted replication**

- Results show that the **potential for latency gains is substantial**
  - Best configuration performs about 38.7% faster on average in terms of observed latency across clients' sites than the median

# Discussion