

# 7CCSMDLC: Distributed Ledgers & Cryptocurrencies

## *Lecture 4: Protocols & Consensus*



**Peter McBurney**

Professor of Computer Science  
Department of Informatics  
King's College London

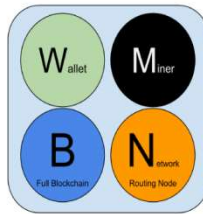
Email: [peter.mcburney@kcl.ac.uk](mailto:peter.mcburney@kcl.ac.uk)

**#2021**



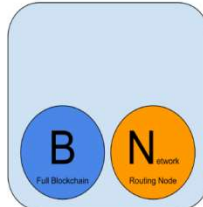
# Outline

- Byzantine Fault Tolerance
- CAP Theorem
- Consensus Protocols



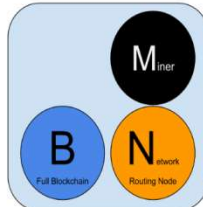
### Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



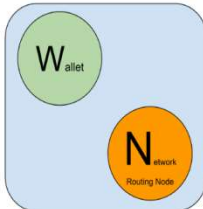
### Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



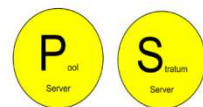
### Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



### Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



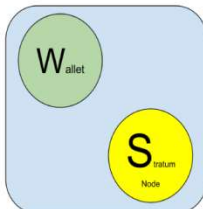
### Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



### Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



### Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.



# Byzantine Faults and CAP Theorem



# Byzantine Generals

- Generals surround a city
  - Each General can Attack or Retreat
  - Success only if **all** Attack or **all** Retreat
- They vote and agree to abide by majority decision
- They are polled and each sends his preference to others
  - 9 Generals: 4 say Attack, 4 say Retreat, 1 is a Traitor
- What happens if the Traitor sends different message to different generals about his vote?
  - He sends *I vote Attack* to the 4 who favour Attack
  - He sends *I vote Retreat* to the 4 who favour Retreat
  - Each group thinks they have a majority
  - Some attack & some retreat
  - Outcome: Failure.

# Bitcoin and the BYZANTINE GENERALS PROBLEM

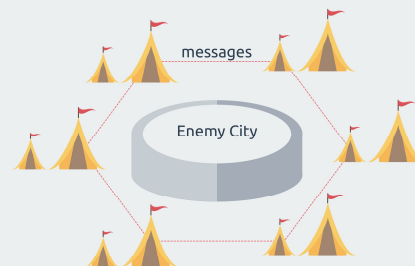
By Patrícia Estevão

## What is the Byzantine Generals Problem (BGP)?



It's a story created to illustrate the difficulty of reaching consensus in distributed systems.

## It begins like this:

An army surrounding an enemy city with various separated camps has to communicate to agree on an attack strategy. There are unknown traitors among this army who can corrupt the message exchange and avoid consensus.



## The Generals and the Bitcoin

	The Situation	
Agree on a Strategy	Objective	Agree on Valid Transactions
Separated Camps	Spacial Distribution	Distributed Nodes in the Network
Loyal Troop and Loyal Generals	The Good Ones	Truthful Nodes
Traitors	The Bad Ones	Evil Nodes
Corrupt a Message	The Attack	Add an Invalid Transaction to the Blockchain
How to Know which Message is True	The Problem	How to know which Transaction is Valid
Don't Have	A Solution	Proof of Work
Don't Have	Consensus	Blockchain with More Combined Difficulty



# Byzantine faults

- Byzantine faults – faults which appear different to different observers
  - A general sends different messages to different colleagues
  - Attack / Retreat
- A malicious node may send different blocks to different other nodes
- Byzantine Fault Tolerance
  - A system designed to withstand such attacks
- Bitcoin blockchain is Byzantine Fault Tolerant
  - Because it makes it costly to send false messages
  - Nodes have to do Work to have a block successfully included in chain.



# Databases on ACID

Desirable properties of a database

- Atomicity
  - Each transaction is all-or-nothing.
  - If part of a transaction fails, then all of it fails.
- Consistency
  - Any data written must be valid according to all rules & constraints
  - Reads of same data must be the same
- Isolation
  - Concurrent execution of transactions must lead to same outcome state as if the transactions were done sequentially
- Durability
  - Once a transaction is committed, it remains so.





# Challenges for large databases

- Relational databases use SQL (Structured Query Language)
- Alternatives to relational databases have come to be known as NoSQL databases
  - Non-tabular data (eg, videos)
  - Flexible schema, disparate data types & formats
  - Distributed and replicated databases
  - Especially for large data holdings
- Examples:
  - Google, Amazon, Facebook, Twitter
- Large databases do not generally support ACID properties
  - Usually Consistency is relaxed
  - Inconsistencies resolved when data is read.

# Example: Apache Cassandra

- Free, open-source NoSQL database management system
  - Originally developed by Facebook, then adopted by the Apache S/W Foundation
- For large data holdings held across multiple servers
- For instance: 2 types of Read transaction:
  - **Single Read:** Search is only local
  - **Quorum Read:** Search is wider and value returned is that endorsed by the majority of nodes after a quorum vote
  - These may return inconsistent results.





# CAP Theorem

- We have a design challenge for distributed databases, expressed by the CAP Theorem:
- CAP Theorem: Only 2 of the following 3 properties are possible to achieve simultaneously:
  - Consistency of data
  - Availability of data
  - Partition-tolerance
- Partition-tolerance refers to how the data is stored or distributed.
- History
  - Conjectured by Eric Brewer in 2000.
  - A limited form was proved by Seth Gilbert & Nancy Lynch.



# Design Implications for Distributed Systems

- Faced with the CAP Theorem, designers cannot forgo partition-tolerance, so must choose between Consistency or Availability.
- This is a trade-off and choice will depend on the requirements of the domain and the use-cases
  - If speed of response is not an issue, then choose Consistency.
  - If response needs to be immediate, then choose Availability.
- In Bitcoin blockchain, the partition arises due to the P2P network.
- The Bitcoin blockchain opts for Availability
  - This may be considered a weakness for a financial system.
  - Although not Consistent, Bitcoin blockchain is **eventually consistent**.



# Bitcoin Blockchain

- There is no master or central node to enforce Consistency.
- So there needs to be a consensus algorithm, for nodes to vote on the true state of the database (the blockchain).
- Bitcoin blockchain is open, so it cannot use simple majority voting
  - No way to ensure that a majority is present
  - No node has overall view of all other nodes
  - A majority could be manipulated (via a Sybil attack)
- Bitcoin blockchain is eventually consistent
  - The older a block is, the more work is required to change it
  - The longest internal fork was 24 blocks in length
    - March 2013, caused by a s/w bug
  - No transaction more than 5 hours old has been reversed.



# Forks in Bitcoin blockchain

There are two types of fork:

- **Regular operations (or internal) fork:** The temporary existence of competing chains as miners process competing blocks
  - Resolved through consensus protocol.
- **Software fork:** In open-source software projects, a fork is a new software project trajectory that starts from an earlier project.
  - Typically, changes to open source software may require a fork.



# Software Forks

Again, there are 2 types of these forks:

- Soft fork: The new version of the software is backwards-compatible
  - In other words, nodes do not need to adopt the new version of the software. In a blockchain, nodes can still recognize new blocks even if they are running the old software version.
  - The network will typically have a mix of nodes running the old & the new software.
- Hard fork: The new version is not backwards compatible.
  - Every nodes needs to adopt the new version of the software to recognize new blocks.

*You run a vegetarian-only restaurant and if you change it to be an vegan-only restaurant then you are soft forking it, if you change it to include meat you are hard forking it.*  
- Andreas Antonopoulos

# Bitcoin hard forks involving currency splits

- Bitcoin Cash:

- Forked at Block 478,558
- 1 August 2017



- Bitcoin Gold:

- Forked at Block 491,407
- 24 October 2017



- Bitcoin Diamond:

- Forked at Block 495,866.
- 12 December 2017.







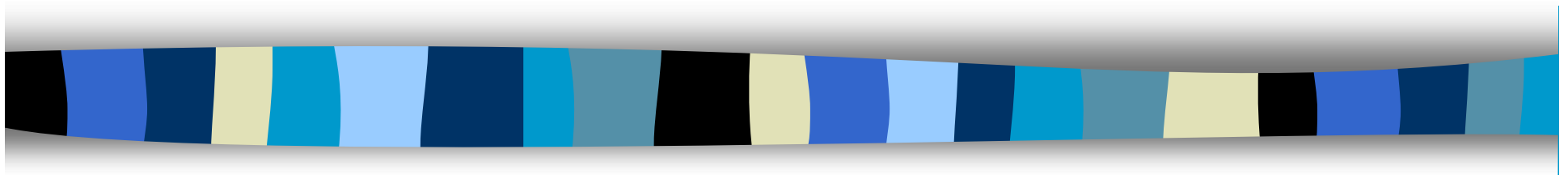
# Altcoins

- Alternative cryptocurrencies
  - Often use code from Bitcoin modified in certain ways
- Some altcoins and intended features:
  - Litecoin (2011): Increased speed of transaction (4 times faster than BTC)
  - Dash (2014): Increased speed of transactions
  - Monero (2014): Enhanced privacy
  - Zcash (Zerocoin) (2016): Enhanced privacy
  - Bitcoin Cash (2017): Increased blocksize
  - Bitcoin Gold (2017): ASIC-resistance.



# Ethereum — The DAO Fork Example

- The DAO: Fund-raise in May 2016
  - Decentralized Autonomous Organization
  - Self-running Venture Capital (VC) fund running over Ethereum
  - Raised \$150 million in 1 month (May 2016) from 11,000 investors
  - Intended that token holders would vote on investment proposals
- June 2016: Vulnerability exploited
  - Not a bug, but exploitation of poor code
  - \$50 million siphoned off
- Ethereum nodes voted to hard-fork to restore lost funds
  - 20 July 2016 at block 1,920,000
  - A form of backwards time-travel
  - 2 branches:
    - Ethereum (forked prior to the exploitation, and the exploit prevented)
    - Ethereum Classic (the exploit continues).



# Consensus Protocols



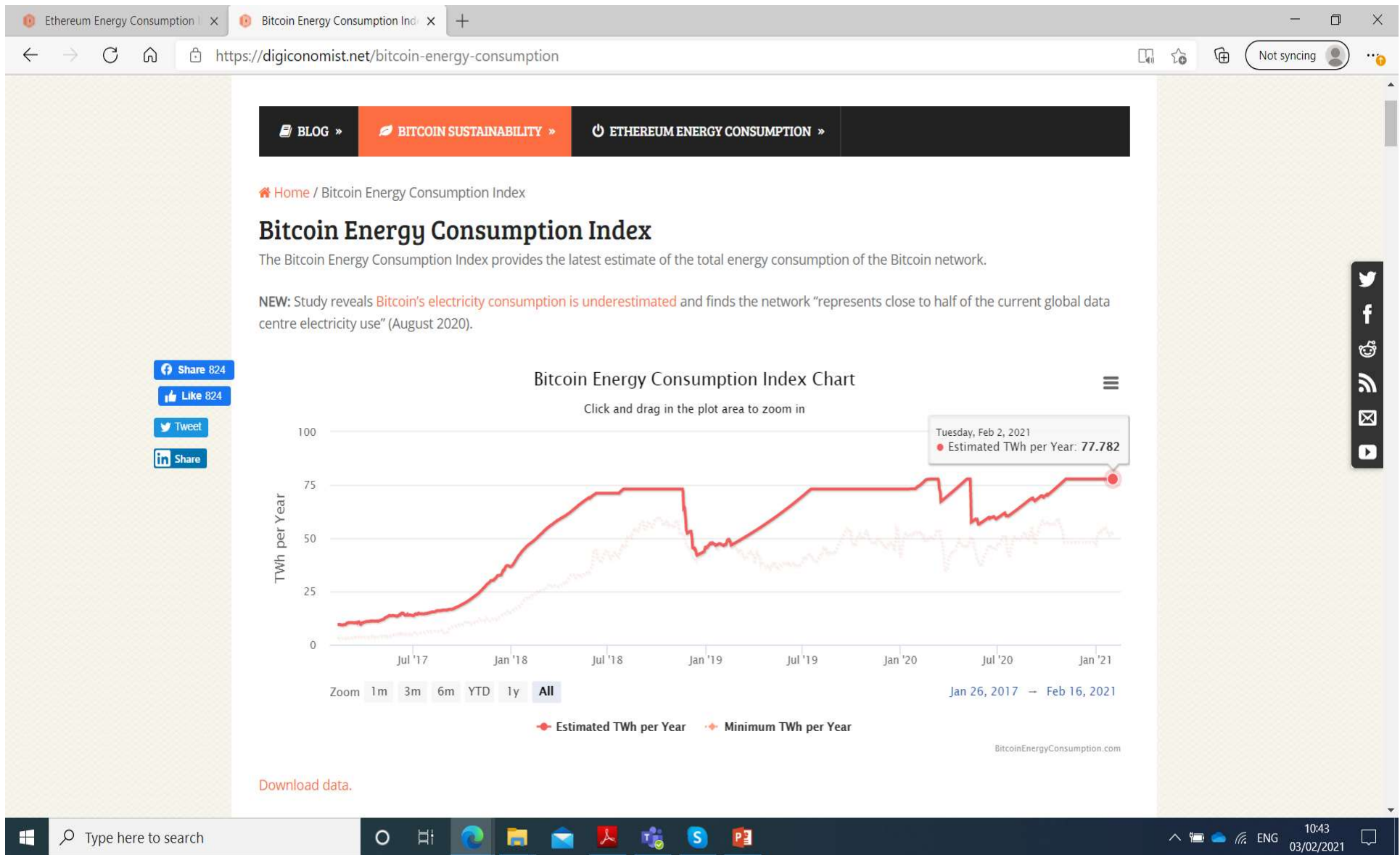
# Consensus Protocols

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Proof of Authority (PoA)
- Other turn-taking protocols

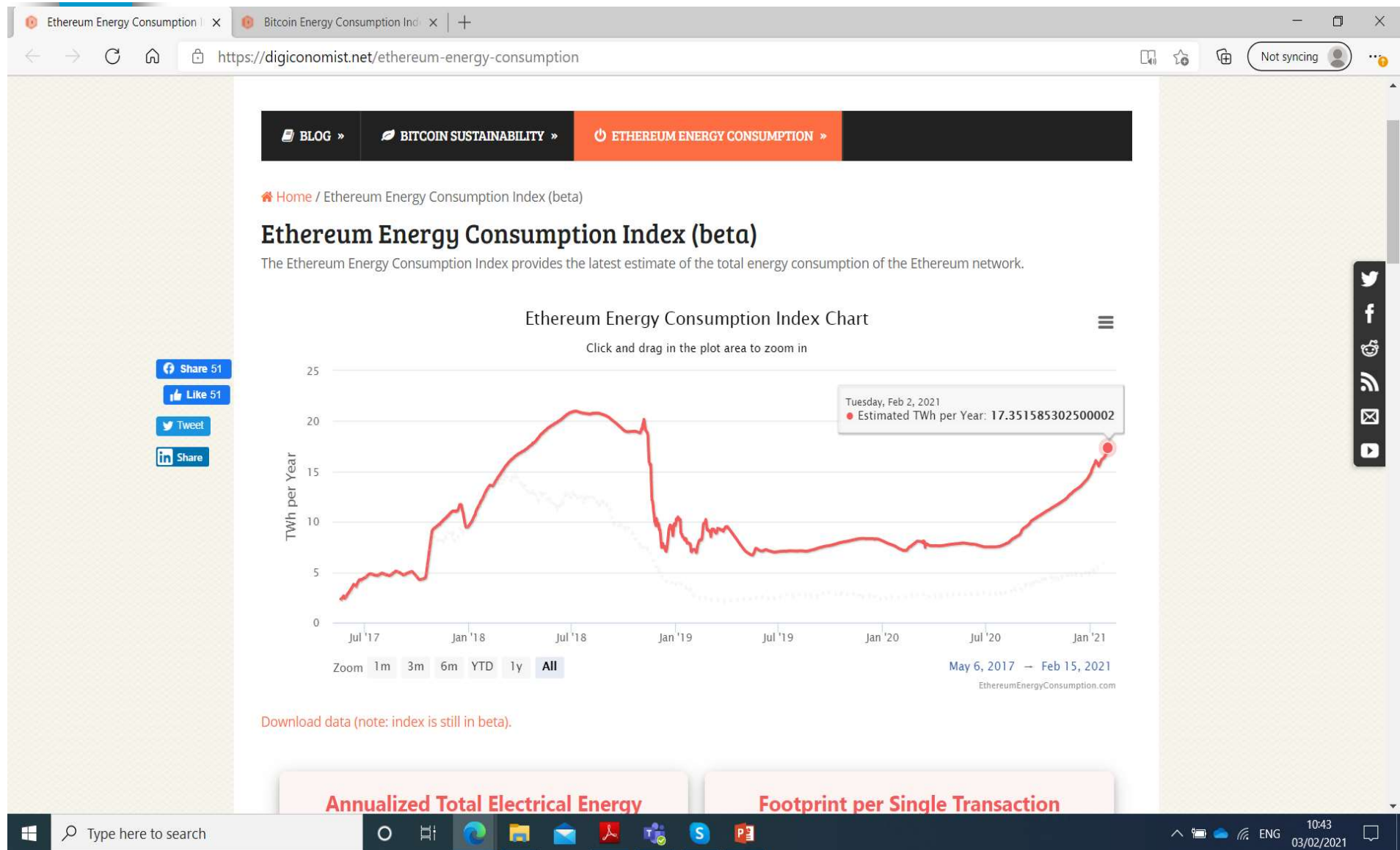


# Proof of Work (PoW)

- How it works
- Why was it proposed?
- Problem of power usage
- Unintended consequences
  - Power usage
  - The use of ASICs and deployment of dedicated compute clusters
  - Large miners & mining pools (coalitions of miners)
  - Concentration of mining power.



Source: digiconomist.net 2021-02-03

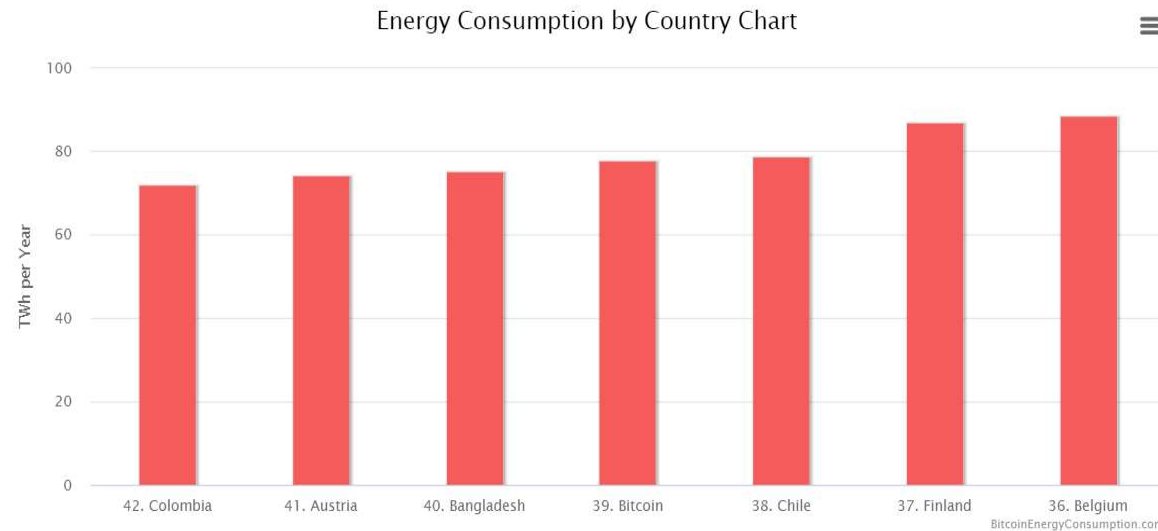


Source: *digiconomist.net* 2021-02-03

# Energy consumption in 2020 vs. nations

Between Bangladesh (161m people) and Chile (19m)

Share 824  
Like 824  
Tweet  
Share



Apart from the previous comparison, it also possible to compare Bitcoin's energy consumption to some of the world's biggest energy consuming nations. The result is shown hereafter.

Source: *digiconomist.net* 2021-02-03





# Proof of Stake (PoS)

- Nodes validate blocks in proportion to their financial stake in the system
  - ie, how much of the internal currency the node has
  - This is a weighted turn-taking protocol (weighted according to stake)
- Then other nodes have to approve
  - Could be a majority of all other nodes (eg, Tendermint)
  - Or a random selection of nodes
- If nodes behave badly, they forfeit their stake
  - No coin creation needed to reward mining, but internal currency still needed
- Examples: Casper, Tendermint
  - Ethereum has transitioned to PoS in Eth 2.0 under the Serenity update
  - Eth2.0 deployed in December 2020, as a parallel chain to Ethereum mainnet.
  - Managing the transition to PoS is a challenge.



## Proof of Stake (2)

- How do design a PoS system so that nodes are economically incentivized to behave well?
  - Cryptoeconomics
  - How to prevent Sybil attacks?
- Need to pay validators only after a block is confirmed
- Vitalik Buterin's Metaphor:
  - 100 people sitting around a table signing competing documents
  - Each person only gets paid if they sign the documents that a majority of other people sign
  - So payment can only be at the end.



# Proof of Stake: Possible flaws

- Initial distribution problem
  - How to incentivize users who have initial coins to transfer coin ownership, since their stake is valuable?
- Validators signing competing blocks
  - This is rational if there is a fork
  - aka Nothing-at-stake problem
- Cartel Problem
  - How to prevent cartels
- Finality Problem
  - How to stop blocks being removed later
  - For PoW systems, the hashing power required to remove a very old block is immense. Is this true in PoS systems?
- If validators forfeit the coin, so what?



# Proof of Authority (PoA)

- Nodes with authority to process blocks are chosen to do so
  - Either chosen randomly or according to some pre-determined rules
  - Nodes may take it in turns to mine blocks
  - Nodes may act as miners for other certain other nodes (eg, as Notaries)
- For example:
  - Companies in a consortium together each take turns to process blocks
- Requires trust in the nodes, and hence only suitable for permissioned ledgers
  - Permissioned ledgers normally have a consortium agreement between the companies involved, which governs investments and penalties
  - There is a business in being an independent notary (eg, BT, NCC Group).
- R3's CORDA platform
  - Uses Notaries to witness & process transactions
  - But this is not strictly a blockchain.



## Other turn-taking protocols

- Protocols could simply decide which nodes should process blocks on some agreed basis, eg
  - Random assignment
  - Time since they last had a turn
- Proof of Elapsed Time (PoET)
  - Developed by Intel as part of their Sawtooth Lake services running on the Hyperledger platform
  - Intended to be run in a secure hardware environment for permissioned ledgers
    - So perhaps less prone to attack by malicious nodes.



# Consensus Attacks

## ■ Sybil attacks

- A malicious node creates proxies that appear to be run by different owners
- In a system using PoW, these would only succeed if the group has sufficient combined hashing power
- They may succeed under other protocols.

## ■ 51% Attacks

- Where a miner or a cartel of miners are able to attack the network because of their combined hashing power
- A majority is not necessarily needed – simulations show that these attacks may succeed with only 1/3 of hashing power (depending on distribution of hash power and structure of network)
- Chinese miners currently could engineer such attacks on Bitcoin.
  - Would it be in their economic interest to do so?
  - Would it be in their strategic political interest to do so?



# Forms of 51% attack

- Create deliberate forks
- Execute DoS (Denial of Service) attacks against specific Transactions or Addresses
  - Just refuse to process these transactions or transactions to/from these addresses
- Double-spend attacks (see next slide).

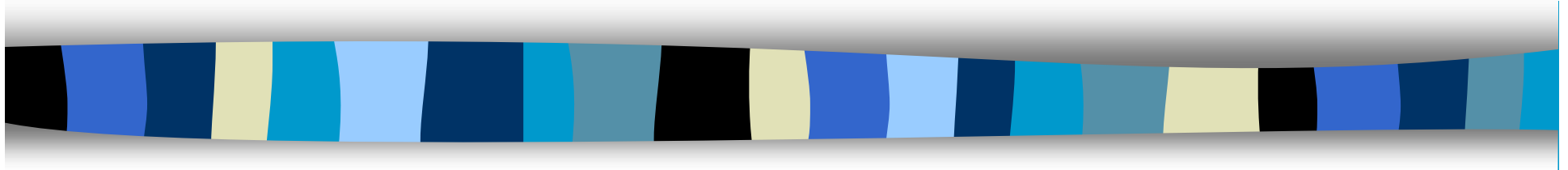


# Double-spend attacks

- How it works:
  - A and C are attackers, B is the mark (victim)
  - A and B have an agreement to exchange bitcoin for some offline goods or fiat money
  - TX1: Attacker A sends bitcoin to B
  - B sees TX1 is confirmed & pays A the offline goods or fiat money
  - TX2: A spends same bitcoin with C (aligned with A)
  - A and colleagues put hash power behind TX2
  - So the blockchain eventually confirms TX2 and not TX1.
- Attackers can only double-spend their own transactions
  - since they have to sign transactions.
- What can B do to avoid loss?
  - Do not accept a single confirmation of TX1 (ie, wait for more blocks)
    - Eg, wait 24 hours (or 144 blocks)
  - Use an escrow (multi-sig) account for the goods/fiat currency.



# Thank you!



[peter.mcburney@kcl.ac.uk](mailto:peter.mcburney@kcl.ac.uk)