Computing for Decentralized Systems

Alejandro Avilés (@OmeGak)

- **Generally, in this series of lectures I'm going to speak about:**
  - Fault tolerance
  - Internet and computers
  - Governance and authority
  - Economics and incentives
  - Blockchain and cryptography
  - Cryptocurrencies and the future

- **Disclaimer:**
  - I don't have a PhD in decentralized systems. I, however, believe the content of these lectures is accurate enough for this audience.

Distributed Systems

Distribute /dɪˈstrɪb.juːt/

To give something out to several people, or to spread or supply something.

- **Examples of distributed systems:**
  - Flight control systems.
  - The Internet (and the networks that compose it)
  - The World Wide Web
  - The Cloud

- **Why using distributed systems?**
  - Sharing resources for increased scalability. Information, printers, idle CPU cycles, etc.
  - Fault tolerance of individual components. What is more likely, one system to fail, or 5 systems to fail?
  - Lower latency. Having the resource closer to where the resource is used.

- **Consequences of distribution:**
  - **Concurrency:** Different computers may do stuff at the same time, sharing resources when necessary and, thus, improving the scalability of the system.
  - **No global clock:** Coordination often depends on a shared idea of the time at which the programs' actions occur but there is no single global notion of the correct time. There is no "now".
  - **Independent failures:** Some or all resources in the network can fail and it is the responsibility of system designers to plan for the consequences of possible failures.

- **Roles-based architectures:**
  - **Client-server:** Client processes interact with individual server processes in potentially separate host computers in order to access the shared resources that they manage.
  - **Peer-to-peer:** Processes interact cooperatively as peers running the same program and offering the same set of interfaces to each other.

- **Examples of client-server interactions:**
  - DNS resolver querying a DNS server.
  - An website loaded in a browser updating its state via AJAX requests to the server.
  - A web server (client) accessing an HTML file stored in the OS file system (server).

- **Examples of p2p applications:**
  - File sharing: Bittorrent, eDonkey.
  - Overlay networks: Invisible Internet Project (I2P).
  - Videoconference: Signal, Jitsi.
  - I'll leave cryptocurrencies for later.

- **Security in distributed systems:**
    - Traditionally, we focused on the threats to distributed systems that arise from the exposure of their communication channels and their interfaces.
    - Encryption can be used to provide adequate protection of shared resources and to keep sensitive information secret when it is transmitted in messages over a network.
    - The portions of a system that are responsible for the implementation of its security and all the hardware and software components upon which they rely have to be trusted or fully-controlled.

Decentralized Systems

Decentralize /ˌdiːˈsen.trə.laɪz/

To move the control of an organization or government from a single place to several smaller ones.

- **A word about the different types of (de)centralization (*):**
    - **Architectural:** The kind of (de)centralization we speak about when we talk about distributed systems.
    - **Political:** The kind of (de)centralization we are going to speak about now.
    - **(*):** The types of decentralization is a topic currently under debate.

- **What is a centralized system?**
  - A centralized system is characterized by the presence of a single agent who:
  - Has complete information about the state of said system.
  - Has complete control to change the state of said system.
  - In complex system, such an agent exerts control over lower-level agents through the use of a power hierarchy.
  - Centralization may be a problem because it creates a social single point of failure.

- **Examples of centralized systems:**
  - A dictatorship.
  - The government of a democratic state.
  - The maintenance of the Linux kernel source.
  - The entire fleet of geographically distributed computers supporting the Google infrastructure.

- **What is a decentralized system? Several definitions:**
  - A system in which agents achieve global goals by coordination by establishing order and coordination by local interactions without a central controller.
  - Self-organization: the spontaneous emergence of order out of a seemingly chaotic ensemble of distributed components.

- **Examples of decentralized systems in nature:**
  - Life itself, securing its self-preservation.
  - An ant colony. No one's in charge. No generals command ant warriors. No managers boss ant workers. The queen plays no role except to lay eggs. Even with half a million ants, a colony functions just fine with no management at all.
  - A beehive. Scout honeybees sense quorum to signal to the rest of the swarm when it's time to move from one hive to another.

- **Examples of decentralized systems in society:**
  - Humanity itself, being each one of us an independent being with free will (*).
  - The voting in general elections in a democratic state (not the counting).
  - International law (if you are a state, that is).
  - A small anarchist community.
  - A large free market economy.
  - The English language.

- **Examples of decentralized systems in computing?**
  - Internet, DNS, Email, XMPP, BitTorrent? Not really.
  - They were built for decentralization, but they ended up centralized in one way or another.
  - Nodes need to be either fully-trusted or fully-controlled, which has lead to centralization.

Consensus and Byzantine Fault Tolerance

Consensus /kənˈsen.səs/

A generally accepted opinion or decision among a group of people.

- **Handling network splits in distributed stateful systems:**
  - Once networks merge back, their nodes need to agree on the state of the system.
  - Strong consistency can be guaranteed with consensus: Paxos, Zab, Raft, etc.
  - These algorithms assume nodes are either trusted or previously controlled by a central agent.

- **The Two Generals' Problem:**
    - Imagine two generals are sieging a city at opposing sides.
    - They both must coordinate an attack, otherwise they will risk being defeated if only one does so.
    - The communication channel, often a messenger, cannot be trusted.
    - One of the general decides to attack and so it informs the other. The one general won't attack until he knows the other has received the message and has acknowledged he will also attack.
    - The other general receives the message that signals the attack and sends back an acknowledgement. This general won't attack until he knows the first general received the acknowledgement.
    - This creates an infinite loop of acknowledgements between the two generals and the decision to attack won't reach finality.
    - Problem described and proved unsolvable in 1975.

- **The Byzantine Generals' Problem:**
    - This is the generalization of the Two Generals' Problem.
    - Now, instead of only 2 generals, there are N generals.
    - To make it worse, generals may be malicious.
- This makes Byzantine faults much harder to deal with than crash faults.
    - Faulty nodes may exhibit arbitrary behavior, which confuses crash-fault detection systems.

- **Why is trust a problem for consensus?**
  - In real life, generals wouldn't be generals if they weren't trustworthy.
  - They do in fact trust each other to some degree.
  - When there are too many generals with which to reach consensus, a hierarchy emerges.
  - Scalability of trust: decentralized O(N^2) vs. centralized O(N).

- **Solutions?**
  - A theoretical solution was proposed in 1999 with the Practical Byzantine Fault Tolerance (PBFT) algorithm.
  - There were no practical implementations of BFT algorithms until Bitcoin came along in 2008 with its Proof-of-Work algorithm.

Governance, Economics, and Proof-of-Work

Governance /ˈgʌv.ɚ.nəns/

The way that organizations or countries are managed at the highest level.

Economics /ˌiː.kəˈnɑː.mɪks/

The social science that studies the production, distribution, and consumption of goods and services focusing on the behaviour and interactions of economic agents.

- **Game theory to solve the Byzantine General's Problem:**
    - The system we need governed is "the siege of the city" in which the economic agents are "the Byzantine generals".
    - We assume the generals are intelligent rational decision-making agents, malicious or not.
    - Generals joining the siege agree on attacking the city through a consensus protocol that involves some sort of voting.
    - Generals have something at stake, the loot of the city in this case.
    - There's no central authority:
        - Generals don't know each other, so nobody knows who's trustworthy.
        - There's no census of generals, so generals could vote infinite times (sybil attack).
        - Voting is, therefore, made expensive, so that generals won't vote infinite times.
    - The game theory model that drives the decisions of general i of the N generals is:
        - reward_i = (totalReward / N) - cost(votes_i)

- **A word of caution:**
    - Such a game theory model will only work if the expected internal reward clearly outmatches the expected external rewards.
    - If so, malicious nodes have a strong incentive to coordinate a successful attack to the consensus system.
    - Decentralization of proof-of-work should not be taken for granted, but should be constantly reevaluated.

- **Hashcash as the first proof-of-work algorithm:**
  - Proof-of-work was invented in 1993 and formalized in 1999.
  - Hashcash algorithm was invented by Adam Back in May 1997.
  - Mechanism to throttle systematic abuse of un-metered internet resources such as email.
  - Used to create stamps to attach to mail to add a micro-cost to sending mail to deter spamming.
  - Vulnerable to sybil attacks, as creation of new identities required for voting are very cheap.
  - Was latter modified for Bitcoin, using CPU rather than identities to cast votes.

Bitcoin and the Blockchain

Bitcoin /ˈbɪt.kɔɪn/

A purely peer-to-peer version of electronic cash that allows online payments to be sent directly from one party to another without going through a financial institution.

- **What is Bitcoin, really?**
  - The Bitcoin network is a bunch of nodes providing a service: increasing the resiliency of a database (blockchain).
  - The Bitcoin nodes can broadcast updates to the state of the blockchain (blocks) as well as validate received updates.
  - The Bitcoin protocol defines rules for checking the validity of state changes.
  - The Bitcoin is an amount of reward given by other nodes after one validly updating the state of the blockchain.
  - Typically, the bulk of data in such database is cryptographically signed transactions of bitcoins between addresses.
  - Finally, transactions can be broadcasted, put in blocks, and protocol rules define when they are valid.

- **Some things to keep in mind with the Bitcoin network:**
    - The service may stop at any point as it runs at a cost (power, cpu cycles, storage, bandwidth, etc.).
    - Anybody can run a node and they may choose not to follow the protocol.
    - People may perceive the bitcoin as a worthless reward.
    - Cryptography may be found broken.

- **So, WHY does the Bitcoin network work, really?**
  - A public, highly resilient database is a valuable service, but one with increasing operational cost.
  - A reward, bitcoins, is granted to nodes that manage to update the state of such database.
  - The validity of state updates depends on the latest valid database state update.
  - To validate state updates, nodes need to keep an up-to-date copy of the database state updates.
  - State updates may contain cryptographically signed reward transactions between addresses.
  - Such database can, therefore, be used as a reward ledger.
  - Bitcoin production is rate-limited and their transactions permission-less, making it a scarce, portable commodity.
  - Bitcoin, thus, gains intrinsic value and people are so incentivized to join the network, run a node, and get profit.

- **Why can we say the Bitcoin network is not only decentralized but Byzantine fault tolerant?**
    - No single agent can prevent somebody updating the state of the blockchain (*).
    - Race conditions are mitigated via block production rate-limiting and resolved via the longest chain rule.
    - Nodes follow rules as producing blocks is made expensive and invalid ones won't be rewarded by the other nodes.
    - As a result, people that don't need to know each other collaborate to provide a trust-less (*), reliable service.
    - (*): Again, only under the assumption internal rewards don't get outmatched by external rewards.

- **Making database state updates (blocks) expensive with proof-of-work (mining):**
  - To verify if a block is valid, its data must be hashed with a double SHA-256 digest.
  - Valid blocks are those whose hash has the required number of leading zero bits (difficulty).
  - For a miner to produce a block, it needs to include in it a set of data that will generate a valid hash.
  - Finding a valid hash involves incrementing an arbitrary number (nonce) and hashing the block until the result valid.
  - This process is computationally expensive and energy intense and, so, a valid block constitutes a proof-of-work.

- **Making rewriting history increasingly expensive with a chain of blocks:**
    - Once the nonce is found, altering any other data in the block makes it invalid and the work needs to be redone.
    - Every block contains a pointer to the previous block and the previous block's hash.
    - Changing and re-mining a previous block alters its hash and, therefore, the content of the next block as well.
    - The next block, thus, is now invalid and it needs to be re-mined also.
    - And the next, and the one after that one, and so on, and so forth...

- **Converging to a single version of the truth with the longest chain rule:**
  - An attacker may want to change one of his own transactions to take back bitcoins they recently spent in goods.
  - To achieve this, they just need to alter the block that contained that transaction and mine all subsequent blocks.
  - There are now two chains: one being extended by honest miners, and another one being extended by the attacker.
  - The further away in the past the tampered block is, the more CPU power the attacker needs to invest to catch up.
  - Because of this, new nodes joining the network always use the longest chain as the reference version of the truth.

- **Summarizing, the security of the blockchain in the Bitcoin network:**
    - As long as the attacker doesn't control the majority of CPU power, the honest chain wont be outrun by the attacker's.
    - The certainty about events occurred in the past increases exponentially with every block added to the blockchain.
    - Strong consistency is gradual and subjective.

Smart Contracts, Ethereum, and Decentralized Applications

Contract /ˈkɑːn.trækt/

A legal document that states and explains a formal agreement between two different people or groups, or the agreement itself.

- **Bitcoin as an alternative application platform:**
    - Bitcoin proved to be a fantastic platform to keep track of money transactions, but not only.
    - **Colored Coins:** Bitcoins with special properties supported by either an issuing agent or by public agreement, and with value independent of the face value of the underlying bitcoins.
    - **Name Coin:** A key/value pair registration and transfer system useful for decentralizing DNS, for instance.
    - **Smart properties:** Proofs of ownership that can be atomically traded but not duplicated.
    - **Smart contracts-ish:** Bitcoin allows some very limited form of scripting that can be used for having digital assets being directly controlled by a piece of code implementing arbitrary rules.

- **But, what is a smart contract?**
  - The contract, a set of promises agreed to in a "meeting of the minds", is the traditional way to formalize a relationship.
  - A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises.
  - The basic idea of smart contracts is that many kinds of contractual clauses can be embedded in hardware and software to make breach of contract expensive (if desired, sometimes prohibitively so) for the breacher.

- **Examples of trust-based smart contracts:**
  - Vending machines.
  - Point of Sale terminals.
  - Electronic Data Interchange, used for ordering and other transactions between large corporations.
  - The SWIFT network, used for transferring and clearing payments between banks.

- **If we were to design (smart) contracts, some basic principles to keep in mind:**
  - Contracts need to be embedded in the world to have effect.
  - Reactive enforcement by threat of physical force is the obvious, traditional way. A judicial system decide what physical steps are to be taken out by an enforcement agency in response to a breach of contract.
  - Proactive enforcement is a physical mechanism that makes breach expensive, such as a combination lock that makes access to a room containing trade secrets expensive without explicit authorization.
  - Reactive enforcement can be minimized by improving proactive enforcement.
  - Smart contracts reference agreements in a dynamic, proactively enforced form, and provide much better transparency where proactive measures may fall short.

- **Decentralizing smart contracts:**
    - Smart contracts often involve trusted third parties, exemplified by an intermediary, who is involved in the performance, and an arbitrator, who is invoked to resolve disputes arising out of performance.
    - Smart contracts in a blockchain can be proactively enforced via consensus and economic incentives rather than relying on third-party trust or a legal framework.
    - Decentralized smart contracts don't make anything possible that was previously impossible, but rather, they allow you to solve common problems in a way that minimizes trust.
    - Minimal trust makes things more convenient by allowing human judgements to be taken out of the loop (*), thus allowing a higher, if not complete, degree of automation.
    - (*): Human judgement still has place in this framework, but I won't be covering it as it's another big topic.

- **Arbitrarily complex smart contracts with Ethereum, a "world computer":**
    - If the Bitcoin network can be understood as a database, Ethereum network would be a general purpose computer.
    - Ethereum is a blockchain with a built-in Turing-complete programming language.
    - Ethereum allows anyone to write smart contracts where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
    - Decentralized applications can call functions in smart contracts to produce changes in the blockchain.

- **Examples of decentralized applications or things that can be created with them:**
  - Crowdsourced knowledge (Gnosis)
  - Decentralized charity (Giveth)
  - Decentralized insurance (Etherisk)
  - Decentralized autonomous organizations (Colony)
  - Decentralized court systems (Aragon)
  - Liquid democracy systems
  - Digital nations?

The Future of Decentralization

Future /ˈfjuː.tʃɚ/

What will happen to someone or something in the time that is to come.

At the beginning of the Internet, people tried to replicate services that existed in the physical world and one of the prime examples is email to postal mail. Nobody could really predict, back then, the applications without counterpart in the physical world that would be built afterwards on top of the Internet, like the Web, BitTorrent, or social networks; nor how they would change people's lives.

Likewise, technologies for decentralized systems are opening the door for implementing many of the services that traditionally required a centralized, trusted party to operate it at scale, but now in a trust-less way. While Internet brought digitalization and almost instant communication, decentralization can potentially remove intermediaries, improve transparency, and make organizations of all sizes flatter and communities fairer.

What lies ahead is unknown and the only way to make it happen is to go build the yet-to-be-imagined decentralized applications that may once more change the the world we live in. Perhaps more than ever before.