Dictatorships, Democracy, and Blockchain

By Mance Harmon

Much has been said about the democratization of technology and how the internet, mobile phones and social media can level the information age playing field. But technology can be an equalizer on a more fundamental level — that of trust. Distributed consensus algorithms enable communities of people, strangers who are both unknown and untrusted, to securely collaborate with each other over the internet without the need for a trusted central server. In the past few years more than a billion dollars have been invested in this type of technology. But why? Why is distributed consensus important? The answer can be found by looking at the differences between a dictatorship and a democracy.

A community can make decisions through either Dictatorship or Democracy. A dictatorship is good if we trust the dictator to not do bad things. A democracy is good if we trust that no large group of members will collude to do bad things.

The trust model of a democracy is arguably superior to that of a dictatorship. In a dictatorship, the group trusts that the dictator makes decisions that are honest and fair. In a democracy, trust is not placed in an individual. Rather, trust is distributed across the population as a whole. Individual members of the group may be unknown and untrusted, but as long as most of the group is trustworthy (honest and fair), then a democracy will result in decisions that are in the best interest of the group.

Today the trust model of computing is usually that of a dictatorship, not a democracy. Software is hosted on servers, and the servers are centrally managed by individuals or organizations. We trust those organizations to provide 1) reliable storage

of information, 2) secure computation, and 3) fair coordination & consensus. For example, we trust Apple and Google to ensure the privacy, integrity, and availability of the information we store in Apple iCloud and Google Drive. We trust Bank of America and Citibank to ensure the accuracy and security of the software that calculates account balances. And we trust the NYSE and NASDAQ to ensure fairness in arbitrating the order of stock trades in those markets.

Placing our trust in centralized systems provides an opportunity for exploitation. Criminals commit cybercrimes by hacking servers. Corrupt governments rig elections. Corporations subvert their online markets by giving preferences to some users over others.

A distributed computing trust model replaces the use of central servers with a community of peers. There is no single server to attack, nor is there a single organization or individual that can unilaterally take actions on behalf of the community. Criminals can't succeed by hacking only a single computer. Rather, an attacker must compromise or control a large portion of the peers, each possibly using a different technology stack (hardware, operating system, applications). Governments cannot rig an election by coercing only a single peer. And Corporations cannot unilaterally break the rules of the system.

To eliminate the use of central servers, we must manufacture trust in a serverless cloud, where each peer may be unknown and untrusted individually. "Manufacturing trust" is having the community vote on what the "server" is doing, where we can trust the result of that election, even if we don't trust the opinion of any particular voter. In other words, we must manufacture trust in a distributed "server" to perform the functions of 1)

reliable storage of information, 2) secure computation, and 3) fair coordination and consensus.

Distributed reliable storage of information and secure computation have existed for some time. However, distributed coordination and consensus of untrusted peers only recently came into widespread use. It enables the development of multi-participant, general-purpose applications that execute without the need for a central server.

Bitcoin's blockchain represents an early step toward manufacturing trust in a peer-to-peer mesh of unknown and untrusted computers. However, significant design choices in current blockchains limit the scope of applications that can be distributed. For example, consider the performance requirements for a notional, distributed game of Minecraft. Let's assume 1000 players, interacting with each other with no central server. Each player might generate 1 or more transactions per second (e.g., moving in the Minecraft world), resulting in a requirement for more than 1000 transactions per second for the community as a whole. Some scenarios will require the community to agree very quickly on the order of transactions. For example, when two players each attempt to pick up the same object, it's important for the community to agree who was first.

The current Bitcoin blockchain has a theoretical maximum throughput of seven transactions per second, far fewer than that required by our notional game. Additionally, it can take an hour for the community to get enough confirmations to agree on which player picked up the object, which is too slow for meaningful game play.

In general, many applications require thousands of transactions per second. Many applications require community agreement on the order of transactions in near real time. Other examples might include distributed office applications like Google Apps for Work (docs, sheets, slides) that can be offered without requiring users to trust Google,

or a distributed eBay that doesn't require a server to enforce the auction rules, or distributed control of SCADA systems to improve system security.

The first generation of distributed consensus algorithms served as a proof of concept of what's possible. The next generation of distributed consensus will enable a cloud of applications, each with a constitution and managed by a democracy, not a dictatorship.