Cosmos Network

ABSTRACT Cosmos is a decentralized network of independent parallel blockchains, each powered by BFT consensus algorithms like Tendermint consensus. In other words, Cosmos is an ecosystem of blockchains that can scale and interoperate with each other. Before Cosmos, blockchains were siloed and unable to communicate with each other. They were hard to build and could only handle a small amount of transactions per second. Cosmos solves these problems with a new technical vision. In order to understand this visi on we need to go back to the fundamentals of blockchain technology.

I.Cosmos and blockchain

A.BITCOIN To understand how Cosmos fits in the blockchain ecosystem, we need to go back to the beginning of the blockchain story. The first blockchain was Bitcoin, a peer-to-peer digital currency created in 2008 that used a novel consensus mechanism known as Proof-of-Work (PoW) It was the first decentralized application on a blockchain. Soon, people started to realize the potential of decentralized applications and the desire to build new ones emerged in the community. At the time, there were two options to develop decentralized applications: either fork the bitcoin codebase or build on top of it. However, the bitcoin codebase was very monolithic; all three layers—networking, consensus and application — were mixed together. Additionally, the Bitcoin scripting language was limited and not user-friendly. There was a need for better tools.

B. ETHEREUM In 2014, Ethereum came in with a new proposition for building decentralized applications. There would be a single blockchain where people would be able to deploy any kind of program. Ethereum achieved this by turning the *Application* layer into a virtual machine called the *Ethereum Virtual Machine (EVM)*. This virtual machine was able to process programs called smart contracts that any developer could deploy to the Ethereum blockchain in a permissionless fashion. This new approach allowed thousands of developers to start building decentralized applications (dApps). However, limitations to this approach soon became apparent and still persist to this day. The first limitation is scaling - decentralized applications built on top of Ethereum are inhibited by a shared rate of 15 transactions per second. This is due to the fact that Ethereum still uses Proof-of-Work and that Ethereum dApps compete for the limited resources of a single blockchain. The second limitation is the relatively low flexibility granted to developers. Because the EVM is a sandbox that needs to accommodate all use cases, it optimizes for the average use case. This means that developers have to make compromises on the design and efficiency of their application (for example, requiring use of the account model in a payments platform where a UTXO model may be preferred). Among other things, they are limited to a few programming languages and cannot implement automatic execution of code. he third limitation is that each application is limited in sovereignty, because they all share the same underlying environment. Essentially, this creates two layers of governance: that of the application, and that of the underlying environment. The former is limited by the latter. If there is a bug in the application, nothing can be done about it without the approval of the governance of the Ethereum platform itself. If the application requires a new feature in the EVM, it again has to rely entirely on the governance of the Ethereum platform to accept it. These limitations are not specific to Ethereum but to all blockchains trying to create a single platform that would fit all use cases. This is where Cosmos comes into play.

C. Cosmos as the next generation of blockchain The vision of Cosmos is to make it easy for developers to build blockchains and break the barriers between blockchains by allowing them to transact with each other. The end goal is to create an Internet of Blockchains, a network of blockchains able to communicate with each other in a decentralized way. With Cosmos, blockchains can maintain sovereignty, process transactions quickly and communicate with other blockchains in the ecosystem, making it optimal for a variety of use cases. This vision is achieved through a set of open source tools like Tendermint, the Cosmos SDK and IBC designed to let people build custom, secure, scalable and interoperable blockchain applications quickly. Let us take a closer look at some of the most important tools in the ecosystem as well as the technical architecture of the Cosmos network. Note that Cosmos is an open source community project initially built by the Tendermint team. Everyone is welcome to build additional tools to enrich the greater developer ecosystem.

II. Tendermint BFT and the ABCI Until recently, building a blockchain required building all three layers (*Networking*, *Consensus*, and *Application*) from the ground up. Ethereum simplified the development of decentralized applications by providing a Virtual-Machine blockchain on which anyone could deploy custom logic in the form of Smart Contracts. However, it did not simplify the development of blockchains themselves. Much like Bitcoin, Go-Ethereum remains a monolithic tech stack that is difficult to fork from and customize. This is where Tendermint, created by Jae Kwon in 2014, came in.Tendermint BFT is a solution that packages the *networking* and *consensus* layers of a blockchain into a generic engine, allowing developers to focus on *application* development as opposed to the complex underlying protocol. As a result, Tendermint saves hundreds of hours of development time. Note that Tendermint also designates the name of the byzantine fault tolerant (BFT) consensus algorithm used within the Tendermint BFT engine. For more on the history of consensus protocols and BFT you can check this cool podcast by Tendermint co-founder Ethan Buchman. The Tendermint BFT engine is connected to the application by a socket protocol called the Application Blockchain Interface (ABCI). This protocol can be wrapped in any programming language, making it possible for developers to choose a language that fits their needs. But Tendermint BFT has some feature that are:

- Public or private blockchain ready: Tendermint BFT only handles networking and consensus for a blockchain, meaning that it helps nodes propagate transactions and validators agree on a set of transactions to append to the blockchain. It is the role of the application layer to define how the validator set is constituted. Developers can therefore build both public and private blockchains on top of the Tendermint BFT engine. If the application defines that validators are elected based on how many tokens they have at stake, then the blockchain can be characterised as Proof-of-Stake (PoS). If however the application defines that only a restricted set of pre-authorized entities can be validators, then the blockchain can be characterised as permissioned or private. Developers have all the freedom to customize the rules that define how the validator set of their blockchain changes.
- High Performance: Tendermint BFT can have a block time on the order of 1 second and handle up to thousands of transactions per second.
- Instant finality: A property of the Tendermint consensus algorithm is instant finality. This means that forks are never created as long as more than a third of the validators are honest (byzantine). Users can be sure their transactions are finalized as soon as a block is created (which is not the case in Proof-of-Work blockchains like Bitcoin and Ethereum).
- Security: Tendermint consensus is not only fault tolerant, it is also accountable. If the blockchain forks, there is a way to determine liability.

III. Cosmos SDK and other application layer frameworks Tendermint BFT reduces the development time of a blockchain from years to weeks, but building a secure ABCI-app from scratch remains a difficult task. This is why the Cosmos SDK exists. The Cosmos SDK is a generalized framework that simplifies the process of building secure blockchain applications on top of Tendermint BFT. It is based on two major principles:

- **Modularity:** The goal of the Cosmos SDK is to create an ecosystem of modules that allows developers to easily spin up application-specific blockchains without having to code each bit of functionality of their application from scratch. Anyone can create a module for the Cosmos SDK, and using ready built modules in your blockchain is as simple as importing them into your application. For example, the Tendermint team is building a set of basic modules that are needed for the Cosmos Hub. These modules can be used by any developer as they build their own application. Additionally, developers can create new modules to customize their application. As the Cosmos network develops, the ecosystem of SDK modules will expand, making it increasingly easier to develop complex blockchain applications.

- **Capabilities-based security:** Capabilities constrain the security boundaries between modules, enabling developers to better reason about the composability of modules and limit the scope of malicious or unexpected interactions. For a deeper look at capabilities click here.The Cosmos SDK also comes with a set of useful developer tools for building command line interfaces (CLI), REST servers and a variety of other commonly used utility libraries. the Cosmos SDK, like all Cosmos tools, is designed to be modular. Today, it allows developers to build on top of Tendermint BFT. However, it can be used with any other consensus engines that implements the ABCI. As time goes by, we expect multiple SDKs to emerge, built with different architecture models and compatible with multiple consensus engines - all within a single ecosystem: the Cosmos Network.

IV. ETHERMINT The great thing about the Cosmos SDK is that its modularity allows developers to port virtually any existing blockchain codebase already in Golang on top of it. For example, Ethermint is a project that ports the Ethereum Virtual Machine into an SDK module. Ethermint works exactly like Ethereum but also benefits from all the properties of Tendermint BFT. All the existing Ethereum tools (Truffle, Metamask, etc.) are compatible with Ethermint and you can port your smart contracts over without additional work.

Why bother creating a blockchain with the Cosmos SDK when I can just deploy my decentralized application on top of a Virtual Machine blockchain?

This question is justified, considering that most decentralized applications today are developed on top of Virtual Machine blockchains like Ethereum. First, it should be stated that the reason for this phenomenon is that up until now blockchains were much more difficult to develop than Smart Contracts. This is not the case anymore, thanks to the Cosmos SDK. Now, developers can easily develop entire application-specific blockchains, which have several advantages. Among others, they give more flexibility, security, performance and sovereignty. To learn more about application-specific blockchains read this post. Of course, if you don't want to build your own blockchain, you can still make your Smart Contracts compatible with Cosmos by deploying them on Ethermint.

V. Connecting Blockchains Together – IBC Now that developers have a way to quickly build customized blockchains, let us see how to connect these blockchains together. The connection between blockchains is achieved through a protocol called Inter-Blockchain Communication protocol (IBC). IBC leverages the instant finality property of Tendermint consensus (although it can work with any "fast-finality" blockchain engine) to allow heterogeneous chains to transfer value (i.e. tokens) or data to each other. IBC allows heterogeneous blockchains to transfer tokens and data to each other, meaning that blockchains with different applications and validator sets are interoperable. For example, it allows public and private blockchains to transfer tokens to each other. Currently, no other blockchain framework enables this level of interoperability.

HOW IBC WORKS?

The principle behind IBC is fairly simple. Let us take an example where an account on chain A wants to send 10 tokens (let us call them ATOM) to chain B. Tracking: Continuously, chain B receives the headers of chain A, and vice versa. This allows each chain to track the validator set of the other. In essence, each chain runs a light-client of the other. Bonding: When the IBC transfer is initiated, the ATOM are locked up (bonded) on chain A. Proof Relay:
Then, a proof that the 10 ATOM are bonded is relayed from chain A to chain B. Validation: The proof is verified on chain B against chain A's header and, if it is valid, then 10 ATOM-vouchers are created on chain B. Note that the ATOM that have been created on chain B are not real ATOM, as ATOM only exist on chain A. They are a representation on B of ATOM from chain A, along with a proof that these ATOM are frozen on chain A. A similar mechanism is used to unlock ATOM when they come back to their origin chain. For a more comprehensive description of the IBC protocol, you can look at this specification.

VII. Designing the "Internet of Blockchains" IBC is a protocol that allows two heterogeneous blockchains to transfer tokens to each other. From there, how do we create a network of blockchains?

One idea is to connect each blockchain in the network with every other via direct IBC connections. The main problem with this approach is that the number of connections in the network grows quadratically with the number of blockchains. If there are 100 blockchains in the network and each needs to maintain an IBC connection with every other, that is 4950 connections. This quickly gets out of hand.

To solve this, Cosmos proposes a modular architecture with two classes of blockchain: Hubs and Zones. Zones are regular heterogenous blockchains and Hubs are blockchains specifically designed to connect Zones together. When a Zone creates an IBC connection with a Hub, it can automatically access (i.e. send to and receive from) every other Zone that is connected to it. As a result, each Zone only needs to establish a limited number of connections with a restricted set of Hubs. Hubs also prevent double spending among Zones. This means that when a Zone receives a token from a Hub, it only needs to trust the origin Zone of this token and the Hub. The first Hub launched in the Cosmos Network is the Cosmos Hub. The Cosmos Hub is a public Proof-of-Stake blockchain whose native staking token is called the ATOM, and where transactions fees will be payable in multiple tokens. The launch of the Hub also marks the launch of the Cosmos network.

VII. convolution Hopefully by now you have a clearer picture of the Cosmos project. Here is a quick recap of what Cosmos is in three concise points:

Cosmos makes blockchains powerful and easy to develop with Tendermint BFT and the modularity of the Cosmos SDK. Cosmos enables blockchains to transfer value with each other through IBC and Peg-Zones, while letting them retain their sovereignty. Cosmos allows blockchain applications to scale to millions of users through horizontal and vertical scalability solutions. More than anything, Cosmos is not a product but an ecosystem built on a set of modular, adaptable and interchangeable tools. Developers are encouraged to join the effort to improve existing tools and create new ones in order to make the promise of blockchain technology a reality. These tools are the foundation needed to create the decentralized internet and global financial system of tomorrow.