[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 0 - Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Blockchain 101](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Blockchain History](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Public and Managed Blockchains](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Consensus in Distributed Networks](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Cryptography](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Self-Assessment Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Go Introduction - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Go Basics](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Go Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Control Structures in Go](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Arrays and Slices in Go](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Standard Packages in Go](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Concurrency in Go](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Docker Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[The Interchain Ecosystem](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Getting ATOM and Staking It](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[A Blockchain App Architecture](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Messages](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Modules](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Multistore and Keepers](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Queries](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Events](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Context](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Testing](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Relaying with IBC](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Interchain Security](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Bridges](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 1 Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 2 - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Setup Your Work Environment](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Run a Node, API, and CLI](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Ignite CLI](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Store Object](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create and Save a Game Properly](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Add a Way to Make a Move](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Emit Game Information](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Record the Game Winner](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 2 Exercise](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[What is IBC?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC Token Transfer](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC Tooling](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[What is CosmJS?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Your First CosmJS Actions](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Compose Complex Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Learn to Integrate Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Put Your Games in Order](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Auto-Expiring Games](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Let Players Set a Wager](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Handle wager payments](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Integration tests](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Incentivize Players](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Help Find a Correct Move](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Understand IBC Denoms](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Go Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Hermes Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create Custom Objects](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Get an External GUI](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Backend Script for Game Indexing](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[IBC Application Developer Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Create a Leaderboard Chain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Run in Production](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Prepare the Software to Run](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Prepare a Validator and Keys](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Prepare Where the Node Starts](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Prepare and Do Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Simulate Production in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Tally Player Info After Production](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Simulate a Migration in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Final Exam](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[What's Next?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

[Continue Your Interchain Journey](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)

Docs Version Switcher

On this page

[What are Tendermint and CometBFT?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-are-tendermint-and-cometbft)

[Consensus with CometBFT and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#consensus-with-cometbft-and-the-interchain)

[Upgradeability of chains](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#upgradeability-of-chains)

[Application Blockchain Interface (ABCI)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#application-blockchain-interface-abci)

[The Cosmos SDK](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-cosmos-sdk)

[State machines](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#state-machines)

[Additional details](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#additional-details)

[CheckTx](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#checktx)

[DeliverTx](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#delivertx)

[BeginBlock and EndBlock](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#beginblock-and-endblock)

[Test yourself - a pseudo-coding exercise](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#test-yourself-a-pseudo-coding-exercise)

[The setup](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-setup)

["Make" the state machine](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#make-the-state-machine)

[What if you like extreme serialization?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-if-you-like-extreme-serialization)

[Overview of upcoming content](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#overview-of-upcoming-content)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#a-blockchain-app-architecture) **A Blockchain App Architecture**



In this section, you will look closer at the application architecture underlying blockchains built with the Cosmos SDK.

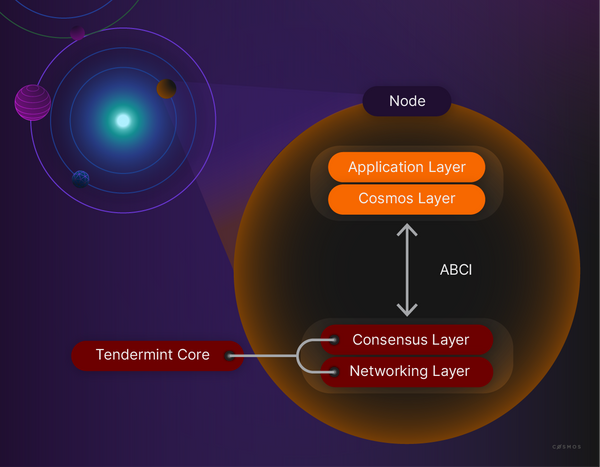
In this section, you will deepen your understanding of the application architecture underlying blockchains built with the Cosmos SDK. You will explore:

* What Tendermint and CometBFT is
* Consensus in the Interchain
* The Application Blockchain Interface
* The Cosmos SDK
* State machines

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-are-tendermint-and-cometbft) What are Tendermint and CometBFT?

Created in 2014, [Tendermint (opens new window)↗](https://tendermint.com/) accelerates the development of distinct blockchains with a ready-made networking and consensus solution, so developers do not have to recreate these features for each new case. You may already use Tendermint without being aware of it, as other blockchains like [Hyperledger Burrow (opens new window)↗](https://hyperledger.github.io/burrow/#/) and the [Binance Chain (opens new window)↗](https://www.binance.org/en/smartChain) use Tendermint.

Tendermint modules **attend to consensus and networking**, which are important components of any blockchain. This frees developers to focus on the application level without descending into lower-level blockchain concerns such as peer discovery, block propagation, consensus, and transaction finalization. Without Tendermint, developers would be forced to build software to address these concerns, which would add additional time, complexity, and cost to the development of their applications.



A blockchain node for an application-focused Cosmos blockchain consists of a state-machine, built with the Cosmos SDK, and the consensus and networking layer, which are handled by [CometBFT (opens new window)↗](https://docs.cometbft.com/v0.37/).

****

**What is CometBFT?**

CometBFT is a blockchain application platform which supports state machines in any language. The language-agnostic CometBFT helps developers securely and consistently replicate deterministic, finite state machines. CometBFT is maintained even when 1/3rd of all machines fail, by providing two components:

* A blockchain consensus engine.
* A generic application interface.

Want to continue exploring this useful component of the Cosmos SDK? Find further information through the [CometBFT documentation (opens new window)](https://docs.cometbft.com/v0.37/).

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#consensus-with-cometbft-and-the-interchain) Consensus with CometBFT and the Interchain

CometBFT is a high-performance, consistent, flexible, and secure **consensus** module with strict fork accountability. It relies on Proof-of-Stake (PoS) with delegation and [Practical Byzantine Fault Tolerance (opens new window)↗](https://arxiv.org/abs/1807.04938). Participants signal support for well-behaved, reliable nodes that create and confirm blocks. Users signal support by staking ATOM, or the native token of the respective chain. Staking bears the possibility of acquiring a share of the network transaction fees, but also the risk of reduced returns or even losses should the supported node become unreliable.

Network participants are incentivized to stake their ATOM with nodes which are the most likely to provide dependable service, and to withdraw their support should those conditions change. A Cosmos blockchain is expected to adjust the validator configuration and continue even in adverse conditions.

Only the top 175 nodes (as ranked by staked ATOM) participate in the transaction finalization process as **validators**. The privilege of creating a block is awarded in proportion to the voting power a validator has. **Voting power** is calculated as all ATOM tokens staked by a validator and its delegates. For example, if a given validator's voting power is 15% of the total voting power of all validators, then the validator can expect to receive the block creation privilege 15% of the time.

The created block is broadcast to the other validators, who are expected to respond promptly and correctly:

* Validators confirm candidate blocks.
* Validators absorb penalties for failing to do so.
* Validators can and must reject invalid blocks.
* Validators accept the block by returning their signature.

When sufficient signatures have been collected by the block creator, the block is finalized and broadcast to the wider network. There is no ambiguity in this process: either a block has the necessary signatures or it does not. If it does, insufficient signatories exist to overturn the block and so the block can be understood as **finalized** - there is no process in which the blockchain would be reorganized. This provides a level of certainty when it comes to transaction finality that a probabilistic system like Proof-of-Work (PoW) cannot match.

Aggressive block times are feasible because the process aims for high performance and is based on dedicated validators with good network connectivity. This is quite different from PoW, which favors inclusion and must accommodate slower nodes with greater latency and less reliability. A Cosmos blockchain can handle thousands of transactions per second, with confirmations taking seven seconds.

Even though validation is delegated to a subset of all network nodes, the process avoids concentration of power. The community of users elects validators by staking ATOM and participating in both the rewards and the risks of committing, to provide a reliable, responsible block validation service.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#upgradeability-of-chains) Upgradeability of chains

In any known blockchain, a change in the implementation requires an upgrade to the node software running on each node. In a disorderly process with voluntary participation, this can result in a hard fork: a situation in which one constituency forges ahead with the old rules while another adopts new rules. While this arrangement has positive aspects and proponents, it also has clear disadvantages in settings where **certainty** is a strict requirement. For example, uncertainty about transaction finality (regardless of the degree of uncertainty) may be unacceptable in settings that are concerned with authoritative registries and large assets.

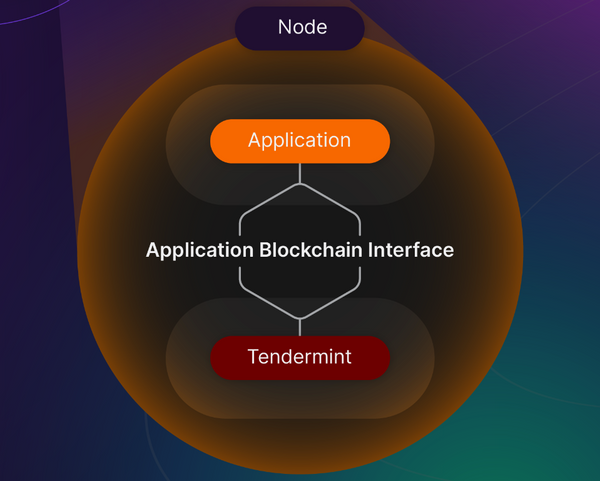
In a CometBFT blockchain, transactions are irreversibly finalized upon block creation, and upgrades are themselves governed by the block creation and validation process. This leaves no room for uncertainty: either the nodes agree to simultaneously upgrade their protocol, or the upgrade proposal fails.

Validators and delegators are the parties who vote on proposals, with weights proportional to their respective stakes. If a delegator does not vote on a proposal, the delegator's vote is taken as that of its delegated validator. This means that delegators should be very demanding when they act, as they also lend their default vote to the validator.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#application-blockchain-interface-abci) Application Blockchain Interface (ABCI)

[CometBFT (opens new window)↗](https://docs.cometbft.com/v0.37/) packages the networking and consensus layers of a blockchain and presents an interface to the application layer, the **Application Blockchain Interface (ABCI)**. Developers can focus on higher-order concerns and delegate peer-discovery, validator selection, staking, upgrades, and consensus to CometBFT. The consensus engine runs in one process and controls the state machine, while the application runs in another process. The architecture is equally appropriate for **private or public blockchains**.

CometBFT is connected to the application by a socket protocol. ABCI provides a socket for applications written in other languages. If the application is written in the same language as the CometBFT implementation, the socket is not used.



CometBFT provides security guarantees, including the following:

* **Forks** are never created, provided that at least half the validators are honest.
* **Strict accountability** for fork creation allows determination of liability.
* Transactions are **finalized** as soon as a block is created.

CometBFT is not concerned with the interpretation of transactions. That occurs at the application layer. CometBFT presents confirmed, well-formed transactions and blocks of transactions agnostically. CometBFT is un-opinionated about the meaning any transactions have.

The *block time* is approximately seven seconds, and blocks may contain thousands of transactions. Transactions are finalized and cannot be overturned as soon as they appear in a block.



For a deeper dive on consensus and CometBFT visit:

* This [podcast on consensus systems (opens new window)↗](https://softwareengineeringdaily.com/2018/03/26/consensus-systems-with-ethan-buchman/) with Ethan Buchman
* The [CometBFT documentation on consensus (opens new window)↗](https://docs.cometbft.com/v0.37/spec/consensus/)

There are at least two broad approaches to **application-level concerns** using blockchains:

1. Create an application-specific blockchain where everything that can be done is defined in the protocol.
2. Create a programmable state machine and push application concerns to a higher level, such as bytecode created by compilers interpreting higher-level languages.

Ethereum-like blockchains are part of the second category: only the state machine is defined in the on-chain protocol, which defines the rules of contract creation, interaction, execution, and little else.

This method is not without its limitations:

* Very little is universally defined: standards for basic concerns such as tokens emerge organically through voluntary participation.
* Contracts can and do contain repetitive code that may or may not correctly implement the developer's intentions.
* This inherent flexibility makes it challenging to reason about what is correct, or even what is friendly.
* There are practical limits to the complexity of operations, which are very low compared to what is possible in other settings.

These limitations make it especially difficult to perform analysis or reorganize data, and developers are forced to adapt to the constraints.

A **purpose-built or application-specific blockchain** is different. There is no need to present a "Turing-complete" language or a general-purpose, programmable state machine because application concerns are addressed by the protocol the developers create.

Developers who have worked with blockchains based on the Ethereum Virtual Machine (EVM) will recognize a shift in the way concerns are addressed. Authorization and access control, the layout of storage and the state, and application governance are not implemented as contracts on a state machine. They instead become properties of a unique blockchain that is built for a singular purpose.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-cosmos-sdk) The Cosmos SDK

Developers create the application layer using the **Cosmos SDK**. The Cosmos SDK provides:

****

**A scaffold to get started**

The Cosmos SDK provides a development head start and a pre-established framework for new applications.

****

**A rich set of modules**

The Cosmos SDK provides a rich set of modules that address common concerns such as governance, tokens, other standards, and interactions with other blockchains through the Inter-Blockchain Communication Protocol (IBC).

The creation of an application-specific blockchain with the Cosmos SDK is largely a process of selecting, configuring, and integrating well-solved modules, also known as "composing modules". This greatly reduces the scope of original development required, as development becomes mostly focused on the truly novel aspects of the application.



The Inter-Blockchain Communication Protocol (IBC) is a common framework for exchanging information between blockchains. For now, it is enough to know that it exists and it enables seamless interaction between blockchains that want to transfer value (token transfers) and exchange information. It enables communication between applications that run on separate application-specific blockchains.

The application, consensus, and network layers are contained within the custom blockchain node that forms the foundation of the custom blockchain.

CometBFT passes confirmed transactions to the application layer through the **Application Blockchain Interface (ABCI)**. The application layer must implement ABCI, which is a socket protocol. CometBFT is unconcerned with the interpretation of transactions, and the application layer can be unconcerned with propagation, broadcast, confirmation, network formation, and other lower-level concerns that CometBFT attends to (unless it wants to inspect such properties).

Developers are free to create blockchains in any language that supports sockets since the ABCI is a socket protocol, provided their application implements ABCI. ABCI defines the boundary between replication concerns and the application, which is a state machine.

This is itself a considerable step forward that simplifies the creation of unique blockchains.

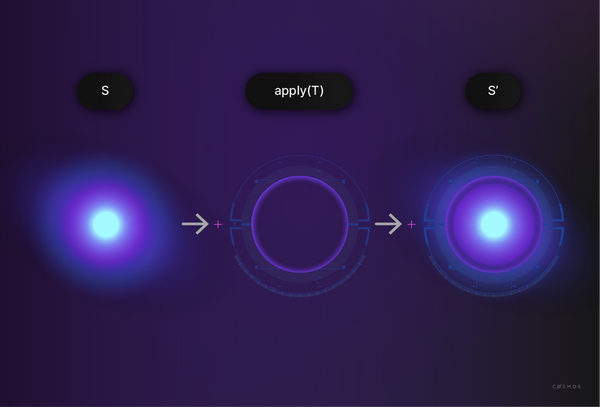


If you want to continue exploring ABCI, you can find more detailed information here:

* [ABCI GitHub repository: ABCI prose specification (opens new window)↗](https://github.com/tendermint/abci/blob/master/specification.md)
* [Tendermint GitHub repository: A Protobuf file on types (opens new window)↗](https://github.com/tendermint/abci/blob/master/types/types.proto)
* [Tendermint GitHub repository: A Go interface (opens new window)↗](https://github.com/tendermint/abci/blob/master/types/application.go)
* [CometBFT documentation (opens new window)↗](https://docs.cometbft.com/v0.37/)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#state-machines) State machines

A blockchain is a replicated state machine at its core. A **state machine** is a computer science concept, in which a machine can have multiple states but only one state at a time. It follows a state transition process (or a set of defined processes), which is the only way the state changes from the old or **initial state (S) to a new state (S')**.

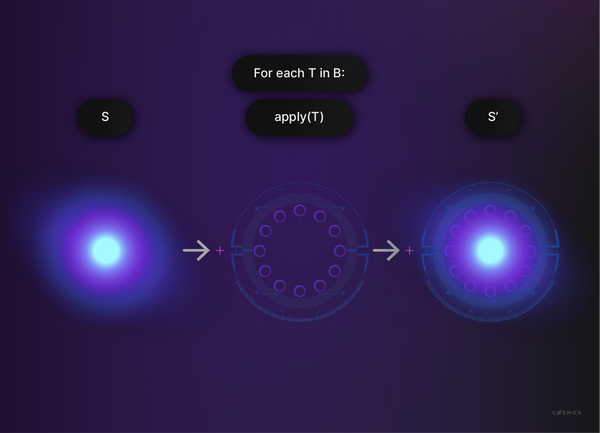


The **state transition function** in a blockchain is synonymous with a transaction. Given an initial state, a confirmed transaction, and a set of rules for interpreting that transaction, the machine transitions to a new state. The rules of interpretation are defined at the application layer.

Blockchains are deterministic. The only correct interpretation of the transaction is the new state, shown as S-prime (S') in the previous image.

Blockchains are distributed, and transactions arrive in batches called blocks. The machine state subsists after the correct interpretation of each transaction in a block. Each transaction executes in the context of the state machine that resulted from every preceding transaction. The machine state after all transactions are executed is a useful checkpoint, especially for historic states.

The state of the initialized blockchain, in which "nothing has happened yet", is called **Genesis state** (S). The current state of the blockchain (S') can always be achieved by applying all the transactions performed to the Genesis state.



Developers can create the state machine using the Cosmos SDK. This includes:

* Storage organization: also known as *the state*.
* State transition functions: these determine what is permissible and if adjustments to the state result from a transaction.

In this context, the "consensus" establishes a canonical set of well-ordered blocks containing well-ordered transactions. All nodes agree that the canonical set is the only relevant set of all finalized transactions. There is only one correct interpretation of the canonical transaction set at any given transaction execution or any block height due to the state machine's determinism.

This state machine definition is silent on the processes that confirm and propagate transactions. CometBFT is agnostic to the interpretation of the blocks it organizes. The Tendermint consensus establishes the ordered set of transactions. The nodes then reach consensus about the state of the application.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#additional-details) Additional details

Transactions and blocks utilize several key methods and message types:

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#checktx) CheckTx

Many transactions that could be broadcast should not be broadcast. Examples include malformed transactions and spam-like artifacts. However, CometBFT cannot determine the transaction interpretation because it is agnostic to it. To address this, the Application Blockchain Interface includes a CheckTx method. CometBFT uses this method to ask the application layer if a transaction is valid. Applications implement this function.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#delivertx) DeliverTx

CometBFT calls the DeliverTx method to pass block information to the application layer for interpretation and possible state machine transition.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#beginblock-and-endblock) BeginBlock and EndBlock

BeginBlock and EndBlock messages are sent through the ABCI even if blocks contain no transactions. This provides positive confirmation of basic connectivity and helps identify time periods with no operations. These methods facilitate the execution of scheduled processes that should always run because they call methods at the application level, where developers define processes.



It is wise to be cautious about adding too much computational weight at the start or completion of each block, as blocks arrive at approximately seven-second intervals. Too much work could slow down your blockchain.

Any application that uses CometBFT for consensus must implement ABCI. You do not have to do this manually, because the Cosmos SDK provides a boilerplate known as **BaseApp** to get you started.

In the following suggested exercise, you will create a minimal distributed state machine with the Cosmos SDK and see code samples implementing concepts progressively. Your state machine will rely on CometBFT for consensus.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#test-yourself-a-pseudo-coding-exercise) Test yourself - a pseudo-coding exercise

With all you have learned about Tendermint and CometBFt, can you **design** a minimal distributed state machine, and think about relevant pseudo-code, for a blockchain that allows people to play the game of checkers? Open the following section, **Creating a checkers blockchain**, to start this reflection and reinforce your understanding of Tendermint.

You will continue to apply what you learn in later sections to your checkers game, and design a blockchain by using elements of the Cosmos SDK. Alternatively, you can continue directly to learn about [accounts in the Cosmos SDK](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html).

****

**Creating a checkers blockchain**

*Why develop a game of checkers?* This **design project** is meant to get you thinking and will evolve in stages as you learn more about Cosmos SDK concepts. You will better understand and experience how the Cosmos SDK improves your productivity by handling the boilerplate as you progress your design through the following sections and explore what the boilerplate does.

This is meant as a design exercise. If you want to go from the design phase to the **implementation** phase, head to [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/), which is a completely separate exercise.

You are free to think about design here and in the following concept pages, and then jump to a related section of the exercise.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-setup) The setup

You are going to design a blockchain that lets people play checkers against each other. There are many versions of checkers, so choose [these simple rules (opens new window)](https://www.ducksters.com/games/checkers_rules.php) for this exercise. The object of the exercise is to understand ABCI and learn more about working with the Cosmos SDK, not to get lost in the proper implementation of the board state or the rules of the game. Use and adapt [this ready-made implementation (opens new window)](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go), including the additional rule that the board is 8x8 and played on black cells. The code will likely require adaptations as you progress. Do not worry about implementing a marketable GUI, that is a separate design project in itself. You must first create the foundation that will ensure a GUI is *possible*.

When you revisit this design exercise in later chapters, the goal will be to improve it with the Cosmos SDK as you learn about its different components. If you are not interested in learning more about ABCI, it is safe to omit *this* exercise and move directly to the other learning elements.

There is a lot you need to do beyond implementing the rules of the game, so simplify as much as possible. Examine these [ABCI specs (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md) to see what the application needs in order to comply with ABCI. Try to identify which basic resources you would use to make a first, *imperfect* checkers game blockchain.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#make-the-state-machine) "Make" the state machine

You want to have a minimum viable ABCI state machine. CometBFT does not concern itself with whether proposed transactions are valid or how the state changes after each transaction. It delegates this to the state machine, which *interprets* transactions as game moves and states. The following are the important junctures at which the application must act:

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#start-the-application) Start the application

This state machine is an application that must start before it can receive any requests from CometBFT. It must load into the memory static elements representing the acceptable general moves. This code exists already and is run automatically when the module is loaded:

Copy

func init()

checkers

checkers.go

[View source](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go" \l "L75" \t "_blank)

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#query-the-game-state) **Query** - ***the*** game state

To simplify, begin with only a single game and a single board. You can decide immediately how to serialize the board without any effort by using this function:

Copy

func (game \*Game) String() string

checkers

checkers.go

[View source](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go" \l "L303" \t "_blank)

Store the board at /store/board and return it in the response's Value when requested via the [Query (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#query-1) command at path = "/store/board". If and when you need to re-instantiate the board state out of its serialized form, call:

Copy

func Parse(s string) (\*Game, error)

checkers

checkers.go

[View source](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go" \l "L331" \t "_blank)

The String() function does not save the .Turn field. You must store whose turn it is to play on your own. Choose a string at /store/turn for the color of the player. Your application needs its own database to store the state. The application must store a state at a certain Merkle root value to be able to recall past states at a later date. This is another implementation *detail* that you must address when creating your application.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#initchain-the-initial-chain-state) **InitChain** - the initial chain state

[This is where (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#initchain) your only game is initialized. CometBFT sends app\_state\_bytes: bytes to your application with the initial (genesis) state of the blockchain. You already know what it would look like to represent a single game. Your application:

* Takes in the initial state.
* Saves it in its database, along with the need for [black (opens new window)](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go#L124) to play next.
* Returns the Merkle root hash corresponding to the genesis state in app\_hash: bytes.

Your application must also handle the list of validators sent by CometBFT. The Cosmos SDK's **BaseApp** will do this.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#a-serialized-transaction) A serialized transaction

Next you must decide how to represent a move. In the ready-made implementation, a position Pos is represented by [two int (opens new window)](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go#L42-L45), and a move by [two Pos (opens new window)](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go#L168). You can decide to represent a serialized move as four int: the first two for the original position (src), and the next two for the destination position (dst).

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#beginblock-a-new-block-is-about-to-be-created) **BeginBlock** - a new block is about to be created

[BeginBlock (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#beginblock) instructs the application to load its state in the right location. Inside header: Header (as per its [detailed definition (opens new window)](https://github.com/tendermint/spec/blob/master/spec/core/data_structures.md#header)) you find:

AppHash: []byte: an arbitrary byte array returned by the application after executing and committing the previous block. This serves as the basis to validate any Merkle proofs that come from the ABCI application, and represents the state of the actual application rather than the state of the blockchain itself. The first block's block.Header.AppHash is given by ResponseInitChain.app\_hash.

This *implementation detail* was skipped before instructing the application to load the right state of the application from its database, which includes the correct /store/board. It is important that the application is able to load a known state at **any** point in time. There could have been a crash or a restore of some sort that de-synchronized CometBFT and the application. The application should work off the last state it has arrived at, in case the header has omitted the AppHash (which should never happen). The application is now ready to respond to the upcoming CheckTx and DeliverTx with the state loaded.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#checktx-a-new-transaction-appears-in-the-transaction-pool) **CheckTx** - a new transaction appears in the transaction pool

CometBFT [asks (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#checktx-1) your application whether the transaction is worth keeping at all. For maximum simplification, you are only concerned with whether there is a valid move in the transaction. You check whether there are four int in the serialized information for this. You can also check that the int themselves are within the boundaries of the board, for example between 0 and 7. It is better **not** to check if the move is valid according to the rules of the application.

Copy

func (game \*Game) ValidMove(src, dst Pos) bool

checkers

checkers.go

[View source](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go" \l "L168" \t "_blank)

Checking whether a move is valid with regards to the board requires knowledge of the board state *when the transaction is included in a block*. The board is updated only up to the point where the transactions have been delivered. You may have a situation where two transactions are sent, one after the other, and both are valid. If you tested the move in the second transaction against the board state before the first unconfirmed move, it would appear that the second move is invalid. Testing a move on the board at CheckTx time should be avoided. Check the *possibility* of validity of the transaction in CheckTx: reject the transaction if it is malformed, contains invalid inputs, etc., and therefore cannot *possibly* be acceptable; but refrain from confirming that it will be successful according to concerns that depend on context.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#delivertx-a-transaction-is-added-and-needs-to-be-processed) **DeliverTx** - a transaction is added and needs to be processed

When a pre-checked transaction [is delivered (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#delivertx-1), it must be applied to the latest board state. You call:

Copy

func (game \*Game) Move(src, dst Pos) (captured Pos, err error)

checkers

checkers.go

[View source](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go" \l "L274" \t "_blank)

Resolve any error if necessary. You need to see whether it makes sense to send the transaction back through ABCI. If the transaction succeeded, you keep the new board state in memory ready for the next delivered transaction. You do not save to the storage at this point. You can also choose to define which information should be indexed via events: repeated Event in the response. The returned values are intended to return information that could be tedious to collect otherwise. This allows fast searches across blocks for values of relevance if indexed.

See [Tendermint's ABCI event spec documentation (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#events) for what goes into an Event.

For the sake of the exercise imagine that you emit some information in two events: one about the move itself, and the other about the resulting board state. In pseudo-code form this looks like:

Copy

[

{ key: "name", value: "moveMetadata", index: true },

{ key: "who-player", value: bool, index: true },

{ key: "is-jump", value: bool, index: false},

{ key: "made-king", value: bool, index: false},

{ key: "has-captured", value: bool, index: false},

{ key: "has-captured-king", value: bool, index: false},

{ key: "is-winning", value: bool, index: true}

],

[

{ key: "name", value: "boardState", index: true },

{ key: "black-count", value: uint32, index: false },

{ key: "black-king-count", value: uint32, index: false },

{ key: "red-count", value: uint32, index: false },

{ key: "red-king-count", value: uint32, index: false }

]

If you come from the Ethereum world, you will recognize these as Solidity *events* with indexed fields that are *topics* in the transaction receipt logs. Unlike Ethereum, though, events are not part of the consensus (the block) but instead are handled purely on the node. It would be judicious to inform CometBFT about the GasUsed (int64). Each move costs the same, so you can return 1.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#endblock-the-block-is-being-finished) **EndBlock** - the block is being finished

Ignoring the issue of validators for now, [this is used (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#endblock) to let CometBFT know what events should be emitted (similarly to how DeliverTx was handled). You have only checkers moves, so it is not very clear what could be interesting at a later date. Assume that you want to tally what happened in the block. You return this aggregate event:

Copy

[

{ key: "name", value: "aggregateAction", index: true },

{ key: "black-captured-count", value: uint32, index: false },

{ key: "black-king-captured-count", value: uint32, index: false },

{ key: "red-captured-count", value: uint32, index: false },

{ key: "red-king-captured-count", value: uint32, index: false }

]

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#commit-your-work-here-is-done) **Commit** - your work here is done

The block is now [confirmed (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#commit). The application needs to save its state to storage, i.e. to its database. The state, which includes /store/board, is uniquely identified by its Merkle root hash. As per the ABCI, this hash has to be returned in a []byte form. For a consensus to be able to emerge, the hash needs to be deterministic after the sequence of the same BeginBlock, the same DeliverTx methods in the same order and the same EndBlock as mentioned in [the documentation (opens new window)](https://github.com/tendermint/tendermint/blob/master/spec/abci/abci.md#determinism). The application may also keep a pointer in its database regarding which state is the latest, so it can purge the board from its memory after having returned and saved. The next BeginBlock will inform the application about which state to load. The application should keep the state in memory to quickly build on it if the next BeginBlock fails to mention AppHash or mentions the same AppHash previously calculated.

[#](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-if-you-like-extreme-serialization) What if you like extreme serialization?

This data does not strictly need to be a Merkle root hash. It could well be any bytes, as long as:

* The result is deterministic and collision-resistant.
* The application can recover the state out of it.

If you took your single board and serialized it differently, you could return the board state as such. You have 64 cells out of which only 32 are being used. Each cell is occupied by either nothing, a black pawn, a black king, a red pawn, or a red king. There are five possibilities, which can easily fit in a byte, so you need 32 bytes to describe the board. Perhaps you can use the very first bit to indicate whose turn it is to play, as the first bit of a byte is never used when counting to five. You now have:

* A deterministic blockchain state.
* Collision-resistance, since the same value indicates an identical state.
* No external database to handle.
* The full state always stored in the block header.



The expandable above is meant as a design exercise. If you want to go from the design phase to the **implementation** phase, head to [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/), which is a completely separate exercise from scratch.   
  
Relevant to this section on ABCI is the [Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/4-game-forfeit.html) section, where you use the Cosmos SDK to implement expiration in EndBlock. Be warned that this is an advanced section of the hands-on-exercise.   
  
In the same vein, and advanced too, is the [Add Leaderboard Module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html) section, where you add an action in EndBlock to reduce computation burdens.

synopsis

To summarize, this section has explored:

* CometBFT, which provides modules that attend to consensus and networking, removing the need for developers to "reinvent the wheel" of lower-level blockchain concerns and freeing them to focus on the application level of their projects.
* CometBFT as a DPoS consensus module with pBFT, through which the top 150 nodes (as ranked by total stake) serve as validators for the blockchain. These duties are divided between validators in proportion to their voting power, and delegating users can assign or withdraw staked ATOM to share in the risks and rewards, and to influence validators in favor of good behavior.
* How CometBFT ensures the upgradeability of chains without the possibility of forking, since (like blocks) any proposed upgrades must be validated by a majority of nodes, and there is no process for reversing validation after the fact.
* The Application Blockchain Interface (ABCI), through which developers are able to work on the higher-order aspects of their projects. The ABCI connects to CometBFT through a socket protocol, and also provides a socket for applications written in other languages, providing additional flexibility to developers.
* The Cosmos SDK, an array of well-solved modular components that developers can rapidly configure and integrate to create the foundations for unique, custom-designed applications.
* The Inter-Blockchain Communication Protocol (IBC), a common framework within the Interchain which allows information exchange between blockchains both inside and outside the Interchain Ecosystem.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#overview-of-upcoming-content) Overview of upcoming content

The following sections will extend your comprehension of the Cosmos SDK and the usefulness of its features. If you went through the previous design exercise, you may have already spotted several shortcomings in your potential game blockchain as it is presently designed:

* Anyone, including the opponent, can post an anonymous transaction and play instead of the intended player. This makes it impossible to know who did what. You need to identify the right player. The Cosmos SDK comes to the rescue with [accounts and signatures](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html).
* You currently have a single game. Multiple games running in parallel would be better, but this would require a well-defined store. Why not explore the Cosmos SDK's [store and keeper](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)?
* It would be good to have an elegant way to serialize data objects of interest and your transactions. [Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html) can help with this.
* How can you penalize spam and bad transactions, and also to be able to play for money? Incorporate tokens defined in another [existing Cosmos SDK module](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html).
* There is a new transaction type: to *create a game*. The Cosmos SDK [context object](https://ida.interchain.io/academy/2-cosmos-concepts/11-context.html) allows you to tailor gas costs according to transaction type.
* If you need to handle validator lists during communication, the Interchain does this out of the box with [BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/8-base-app.html).
* Do you want the player's GUI to easily reload any pending games, or let them know whether a move is valid or not? These are good uses of Cosmos SDK [queries](https://ida.interchain.io/academy/2-cosmos-concepts/9-queries.html).
* If you want to use CometBFT to notify players when it's their turn, Cosmos SDK provides that option with [events](https://ida.interchain.io/academy/2-cosmos-concepts/10-events.html).
* What if you want to add changes to your system in the future after production? You can easily handle this with Cosmos SDK [migrations](https://ida.interchain.io/academy/2-cosmos-concepts/16-migrations.html).

previous

[](https://ida.interchain.io/academy/1-what-is-cosmos/3-atom-staking.html)

**[Getting ATOM and Staking It](https://ida.interchain.io/academy/1-what-is-cosmos/3-atom-staking.html)**

up next

**[Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html)**

[[](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html)](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[What are Tendermint and CometBFT?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-are-tendermint-and-cometbft)

[Consensus with CometBFT and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#consensus-with-cometbft-and-the-interchain)

[Upgradeability of chains](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#upgradeability-of-chains)

[Application Blockchain Interface (ABCI)](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#application-blockchain-interface-abci)

[The Cosmos SDK](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-cosmos-sdk)

[State machines](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#state-machines)

[Additional details](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#additional-details)

[CheckTx](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#checktx)

[DeliverTx](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#delivertx)

[BeginBlock and EndBlock](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#beginblock-and-endblock)

[Test yourself - a pseudo-coding exercise](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#test-yourself-a-pseudo-coding-exercise)

[The setup](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#the-setup)

["Make" the state machine](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#make-the-state-machine)

[What if you like extreme serialization?](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#what-if-you-like-extreme-serialization)

[Overview of upcoming content](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html#overview-of-upcoming-content)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)