Hyperledger Sawtooth for Application Developers

Module 3: Sawtooth Overview

MODULE 3: Sawtooth Overview

This module introduces you to Hyperledger Sawtooth, an enterprise-level blockchain platform.

Contents

- What is the Hyperledger Foundation?
- What is Hyperledger Sawtooth?
- Sawtooth Features
- Sawtooth Architecture
  - Sawtooth core, transaction processors, and client apps
  - Sawtooth network
  - Validator
  - About consensus
- Sawtooth Applications
  - Application components: Data model, transaction processor, REST API, and client app
  - Transactions, batches, and blocks
  - State and addressing
  - Permissioning and security
- Event Subscription

What is the Hyperledger Foundation?

The Hyperledger Foundation

The Hyperledger Foundation is "an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chain, manufacturing and technology."

The Hyperledger Foundation focuses on enterprise-grade blockchain projects, including Hyperledger Sawtooth. For more information, see hyperledger.org.

What is Hyperledger Sawtooth?

Hyperledger Sawtooth is an enterprise solution for building, deploying, and running distributed ledgers. It provides a modular and flexible platform for implementing transaction-based updates to shared state (the blockchain) between untrusted participants. Approval to make these updates is coordinated by consensus algorithms.

Sawtooth simplifies blockchain application development by separating the core system from the application domain. Application developers can specify the business rules appropriate for their application, using the language of their choice, without needing to know the underlying design of the core system.

Sawtooth is also highly modular. The Sawtooth architecture allows applications to choose the transaction rules, permissioning, and consensus algorithms that support their unique business needs.

For more information, see the Hyperledger Sawtooth project on hyperledger.org.

Sawtooth Features

Sawtooth has several distinctive features:

- **Separation between the application level and the core system**
  Sawtooth's modular design lets you focus on business rules without needing to know the underlying design of the system. *Transaction processors* handle the application's server-side business logic, while *validators* manage the core functions of verifying transactions and reaching consensus. Transaction rules, permissioning, and consensus settings are handled in the transaction-processing layer.

- **Global state agreement**
  Sawtooth offers cryptographically verifiable distributed ledgers with global state agreement, which is an assurance that each node has cryptographically identical copies of the blockchain database.

- **Dynamic consensus algorithms**
  Consensus is the process of building agreement among a decentralized group of mutually distrusting participants. Sawtooth isolates consensus from transaction semantics. More importantly, Sawtooth allows different types of consensus on the same blockchain. The consensus is selected during the initial network setup and can be changed on a running blockchain with a transaction. Consensus is modular as it is implemented as a consensus engine in a separate process.

- **Parallel transaction execution**
  Most blockchains require serial transaction execution to guarantee consistent ordering at each node on the network. Sawtooth includes an advanced parallel scheduler that splits transactions into parallel flows. Based on the locations in state that are accessed by a transaction, Sawtooth isolates the execution of transactions from one another while maintaining contextual changes.

Sawtooth Features (continued)

- **Multi-language support**
  Sawtooth allows blockchain applications to be written in any language. Further, it includes SDKs for a variety of languages, including Python, JavaScript, and Go. You can use different languages for client-side and server-side code. The variety of SDKs lets you select the technology and languages that are best for your application.

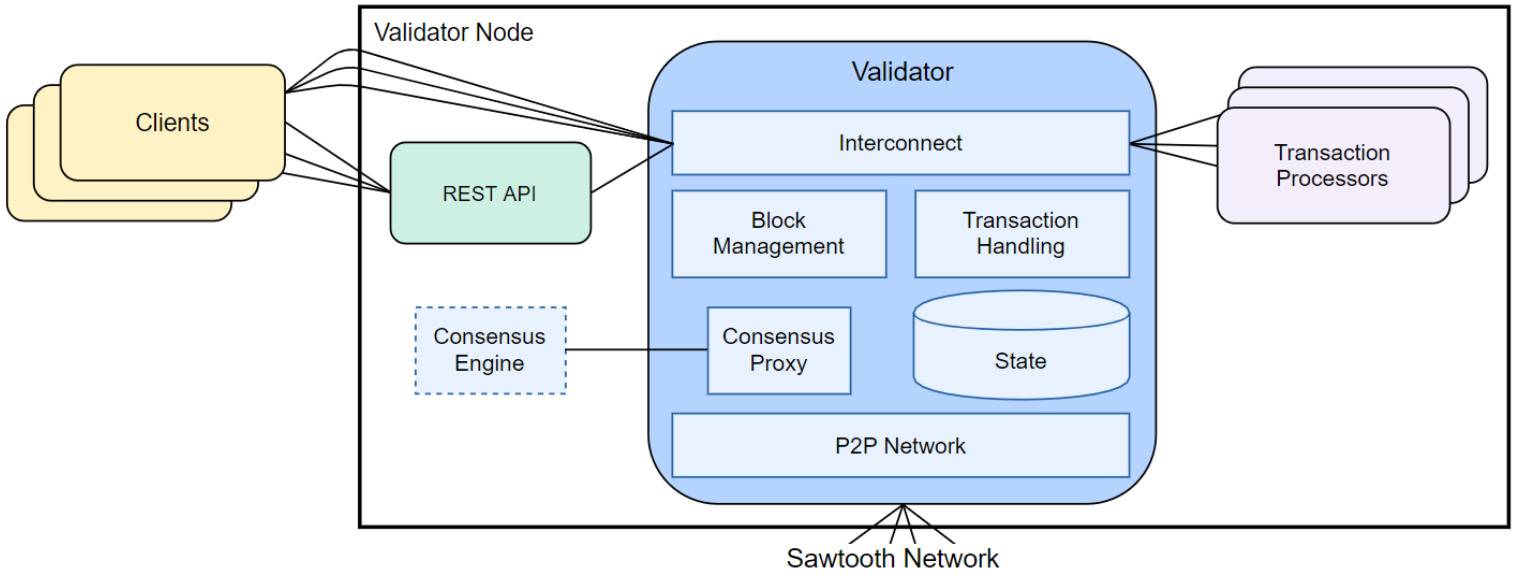
- **An event system that supports creating and broadcasting events**
  Sawtooth applications can:
  - Subscribe to blockchain-related events, such as committing a new block or switching to a new fork.
  - Subscribe to application specific events defined by a transaction family.
  - Relay information about transaction execution to clients without storing that data in state.

- **On-chain governance for permissions and configuration setting**s
  Private networks of Sawtooth nodes can be easily deployed with on-chain permissioning. The blockchain stores configuration and permission settings, such as roles and identities, so that all participants in the network can access this information.

- **Ability to integrate with other blockchain technologies**
  The Sawtooth Ethereum (Seth) project is an initial proof-of-concept integration between the Hyperledger Sawtooth and Hyperledger Burrow projects. With Seth, EVM smart contracts can be deployed to Sawtooth. This course does not cover Seth, but you can learn more from the Hyperledger blog post Hello World, Meet Seth.

Sawtooth Architecture

Sawtooth separates core blockchain functions from the business logic through the use of transaction processors and client applications.



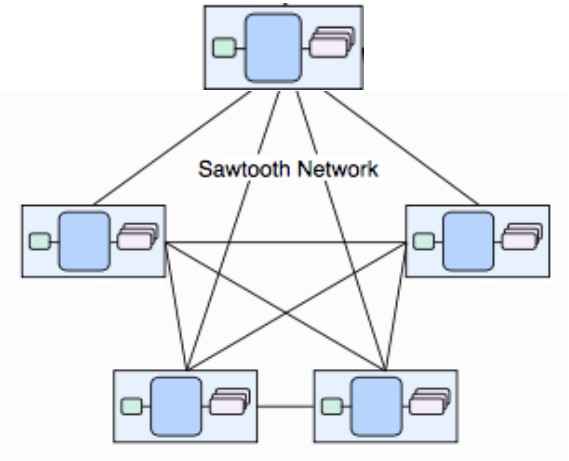
Sawtooth core:
- Message handling
- Block validation & publishing
- Global state management
- Modular consensus engine
- Transaction processing (serial or parallel)

Transaction processor:
- Transaction validation
- Block creation
- Transaction rules (business logic)

Client:
- Transaction generation (payload)
- Data display
- Event handling: Reacting to state changes, failure to commit a block, forks, etc.

Sawtooth Network

A Sawtooth network consists of a set of validator nodes. A validator node is a host system (physical computer, virtual machine, or set of Docker containers) that runs a validator process and an identical set of transaction processors.



The genesis block is created for the first validator node only. It includes on-chain configuration settings, such as the consensus type, that will be available to the new validator nodes once they join the network.

★ The first validator node on the network has no special meaning, other than being the node that created the genesis block (the first block on the blockchain). Sawtooth has no concept of a "head node" or "master node". In a Sawtooth network, each node has the same genesis block and treats all other nodes as peers.

Validator

A Sawtooth validator is a core component that does the following:

- Validates batches of transactions
- Combines batches into blocks
- Maintains consensus with the Sawtooth network for adding candidate blocks to each node's version of the blockchain
- Coordinates communication between clients, transaction processors, and other validator nodes

Each Sawtooth node runs one validator process. Each validator has its own instance of the blockchain and communicates with the other validators using a peer-to-peer network. The gossip protocol enables communication between the validators.

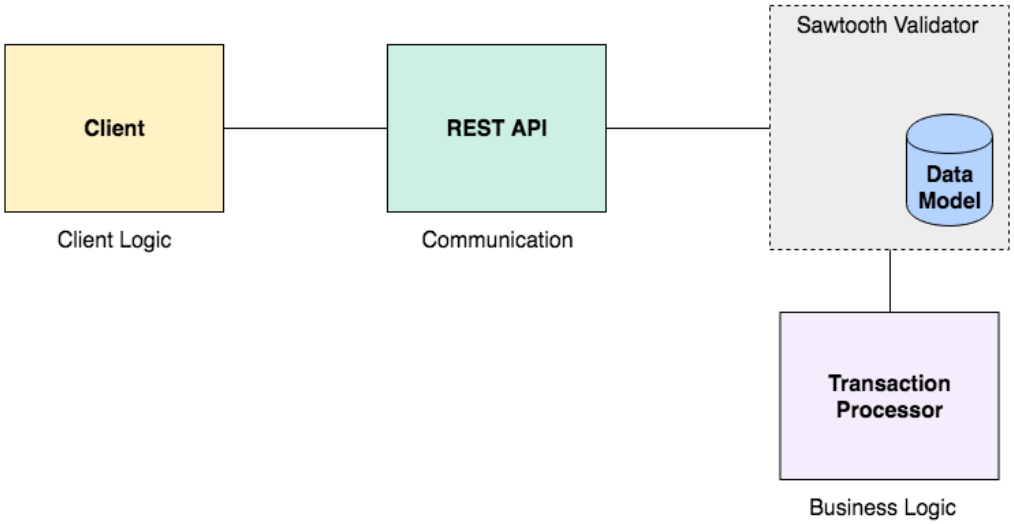
About Consensus

Sawtooth supports *dynamic consensus*, which can allow different types of consensus on the same blockchain. Dynamic consensus means the consensus algorithm can be changed for a blockchain without reinitializing the blockchain or restarting Sawtooth. Consensus is modular as each consensus algorithm it is implemented as a separate module running as a separate process. Sawtooth currently supports these consensus implementations:

- Proof of Elapsed Time (PoET), a Nakamoto-style consensus algorithm that is designed to be a production-grade protocol capable of supporting large network populations. PoET relies on secure instruction execution to achieve the scaling benefits of a Nakamoto-style consensus algorithm without the power consumption drawbacks of the Proof of Work algorithm.
- PoET simulator, which provides PoET-style consensus on any type of hardware, including a virtualized cloud environment.
- PBFT, a leader-based, non-forking consensus algorithm with finality that provides Byzantine Fault Tolerance (BFT). Ideal for smaller, consortium-style networks that do not require open membership.
- Dev mode, a simplified random-leader algorithm that is useful for development and testing.

Sawtooth Applications

Hyperledger Sawtooth separates the application level from the core level, so that an application's business logic is independent from the core Sawtooth functions.



An application can be as simple as a single transaction format with associated validation and state-update logic, or as complex as a virtual machine with opcode accounting and bytecode stored in state (on the blockchain) as *smart contracts*. The application defines the operations or transaction types that are allowed on the blockchain.

Sawtooth includes example applications (called "transaction families" in the documentation) to serve as models for low-level functions, such as maintaining chain-wide settings and storing on-chain permissions, and for specific applications, such as performance analysis and storing block information.

Transaction processor SDKs are available in multiple languages, including Python, JavaScript, Go, C++, Java, Swift, and Rust, to streamline the creation of new applications.

Sawtooth Application Components

A Sawtooth application includes these components:

- A data model to define valid operations and specify the transaction payload (data for the application).
- A **transaction processor** to define the business logic for your application. The transaction processor validates batches of transactions and updates state based on the rules defined by the application. The transaction processor runs on each validator node in the Sawtooth network.
- A **client** to handle the client logic for your application. The client generates and sends transactions (changes for the blockchain) to the validator; the client also displays data. A client can run on a separate system or on the validator node. For example, Sawtooth includes a set of commands that run as a client on each validator node.
- A **REST API** (optional) to communicate between the client and the transaction processor. An application can use a custom REST API, or Sawtooth provides a REST API that simplifies client development by adapting client-validator communication to standard HTTP/JSON. The REST API runs as a service on the validator node or a separate system.

★ A Sawtooth application is called a *transaction family* in the Sawtooth documentation. A transaction family defines the data model and the set of possible transactions for an application.

The rest of this section describes the structure of Sawtooth transactions, batches, and blocks, explains how Sawtooth handles state and addressing, and summarizes the consensus types.

Transactions, Batches, and Blocks

State changes occur when the Sawtooth validator commits a *block* containing a *batch* of *transactions*. First, a client creates a transaction, wraps it in a batch, and submits the batch to the validator (usually via a REST API). The validator verifies the batch, wraps it in a block, and commits it to the blockchain. At that point, the transactions in the block cause the blockchain state to change.

- **Transaction**: A single change in the state of the blockchain. Each Sawtooth transaction includes a transaction header that identifies the signer (the client that created the transaction) and a unique transaction ID.
- **Batch**: A transaction is always wrapped inside of a Sawtooth batch, which is the atomic unit of state change. All transactions within a batch are either committed to state together or are not committed at all. Each Sawtooth batch has a batch header that includes the public key of the client who created the batch (often the same as the signer) and the application name (transaction family name) and version. This feature means that a transaction does not have to declare explicit dependencies. Also, transactions from different applications can be combined in a batch. For example, transactions relating to on-chain settings could be batched with application-specific transactions.
- **Block**: A group of Sawtooth batches. A block has a header that includes a timestamp, a signer, and a hash (unique block ID). After a block is committed, the header also identifies the previous block in the blockchain.

★ Transactions and batches can be signed by different keys. For example, a browser application can sign the transaction and a server-side component can add transactions, then create and sign the batch. This feature allows an application to aggregate transactions from multiple transactors into a single batch operation.

State is the Sawtooth blockchain, represented as a log of all changes that have occurred since the genesis block. Each validator node has its own copy of state in a local database. As blocks are published, each validator updates its state database to reflect the current blockchain status.

Global state (or *global state agreement*) represents the consensus of all participants on the state of the blockchain.

Sawtooth represents blockchain state as a tree on each validator. State is split into application-specific namespaces (addresses), so that each application knows which portion of state it is allowed to change. Also, application developers can define, share, and reuse global state data between applications.

The permissioning design for Hyperledger Sawtooth includes:

- Transactor and batch key permissioning, which controls acceptance of transactions and batches based on signing keys.
- Validator key permissioning, which controls which nodes are allowed to establish connections to the validator network.
- Policy enforcement: A set of DENY and PERMIT rules that can be used to control access to the validator network and determine which transactors can participate on the network.

Sawtooth Events

A Sawtooth application can subscribe to core Sawtooth events and application-specific events that occur on the blockchain, such as a new block being committed or switching to a new fork, then inform clients about the transaction execution without storing that data in state.

An application can also request *event catch-up* by issuing a catch-up subscription request with a specific block ID. The Sawtooth validator sends data for all events that have occurred after that block was committed.

An event subscription includes the event type, an address (as a specific address, a range, or a pattern), and optional filters for event attributes. Most event subscriptions use filters to focus on the specific events of interest.

Later modules will explain how to implement this functionality in the example Sawtooth Simple Supply application, which includes an event subscriber and a reporting database.

Module 3: Summary

This module described the key concepts of Hyperledger Sawtooth, including the separation of core functionality from the application level, Sawtooth batches, and event subscription.

The next module explains the fundamental issues of Sawtooth application design.

Two Chains, Roman
J. Paul Getty Museum
Digital image courtesy of the Getty's Open Content Program.