EOSBake

An EOS.IO Infrastructure Platform for Sidechain Deployment

V0.1 EOS.IO

1. Introduction

This White Paper describes the core concepts behind EOSBake an EOS.IO Infrastructure Platform, implemented by Global Crypto Holdings.

Global Crypto Holdings Inc. is an InfoSec, Fintech and Consulting company focused on innovative Blockchain solutions for the financial service industry. Global Crypto Holdings (GCH) specializes in emerging crypto capital markets, insurance, digital asset security and regulatory compliance.

Real world institutions are usually organizations that pay for infrastructure and operational maintenance of groups of human individuals that operate largely as mere computational resources. Human limitations lead to paperwork, which leads to costs, bureaucracy and inefficiency. End-users are usually forced to pay in order to use the platform or benefit from

its services and cover the costs. The payment usually occurs in financial assets or plain consumer attention for advertisements. Modern state and corporate institutions are suffering a deep democracy crisis where citizens feel their concerns are not being properly represented. New technologies promise to leverage the EOS.IO Governed Blockchain to mitigate these issues.

A Decentralized Autonomous Community (DAC) is a trustless organization that is run by a set of rules encoded as smart contracts and a constitution. A DAC is expected to run with little to no human interference, provided that the smart contracts are executed by a Turing Complete platform. The DAC's transaction record and programmatic rules are stored and executed on a virtual machine/database that operates as a virtual Operating System. A DAC presents itself to society as a means to provide a secure digital ledger to track asset interactions across the internet. It mitigates forgery with trusted timestamping and a distributed database. This eliminates the need for a mutually accepted trusted third party in asset transactions. Infrastructure costs are drastically reduced by the elimination of the trusted third party and the inherent bureaucracy of traditional institutions. Human beings are promoted from the obsolete bureaucracy duties to the actual complex decision making and voting processes.

The concept of a DAC was first proposed in the article "Overpaying for Security", by Daniel Larimer. Since 2014 many Decentralized Autonomous Organizations (DAO) have been implemented in the Ethereum platform.

EOSBake provides an EOS.IO Blockchain Infrastructure Platform that enables stakeholders to set up their own EOS.IO Blockchain infrastructure (hardware+software) in a trusted and auditable way.

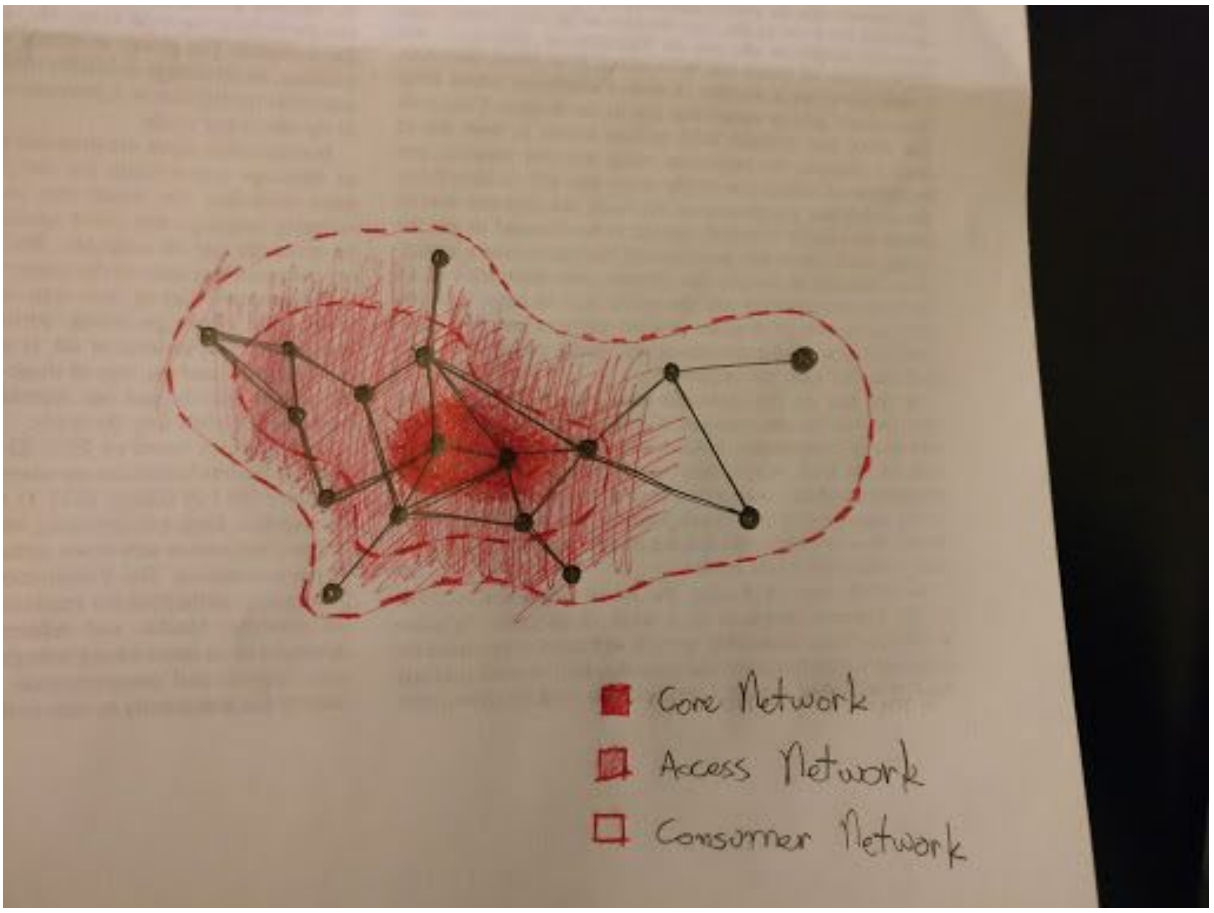
Relying on third party Cloud Service providers is not an ideal long term solution for DACs, because this approach ultimately leads to centralization. Stakeholders must be able to have full control of the hardware and the whole software stack underlying the nodes of the blockchain infrastructure.

The following sections discuss concepts such as Governance, EOS.IO Blockchain Architecture, Network Infrastructure, and Automated Build Systems.

2. Blockchain Infrastructure

1.1. EOS.IO

A full-scale EOS.IO network is formed by thousands of servers distributed worldwide performing a variety of functions. The EOS.IO network can be visualized as concentric layers, with the innermost layer as the Core Network, surrounded by the Access Network, which is accessed by a global community of users through the Consumer Network.



The boundaries between network layers are not hardly defined. The location of a node in a particular network is defined in a fluid manner. Depending on community consensus, nodes can move between layers or reside in multiple layers.

There is no single owner enforcing any boundaries. Participants can come from different and independent organizations within the community. Participants can position or lobby for position in any of the layers. The community collectively decides who plays each role.

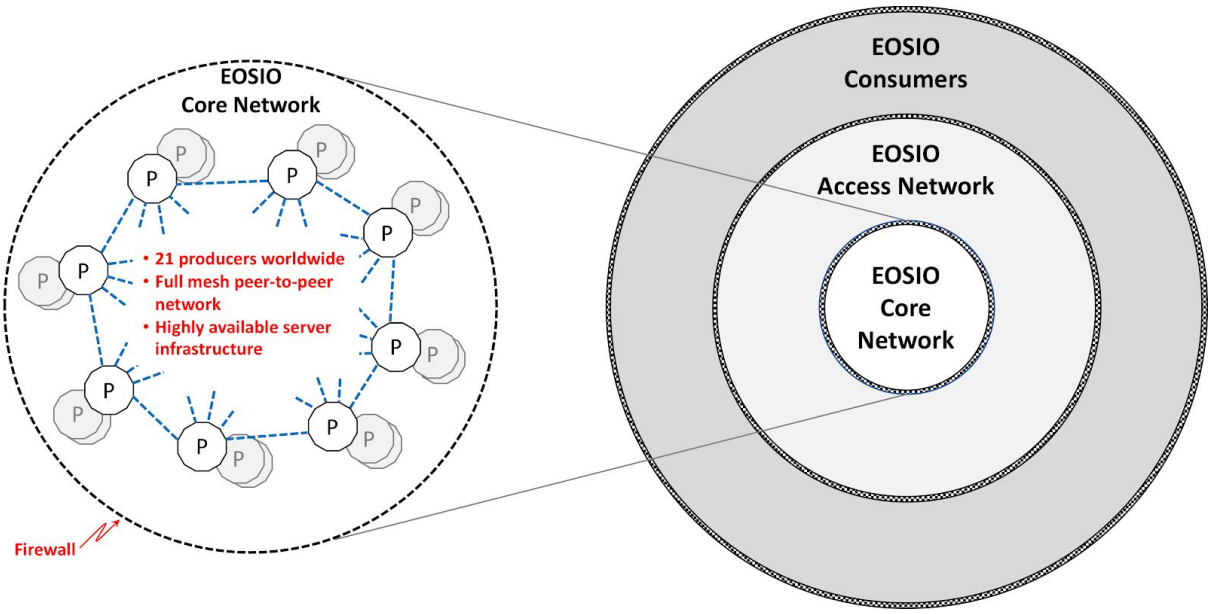
1.1.1. EOS.IO Core Network

The core of the EOS.IO network is made of block producers and the necessary server infrastructure. A full-scale EOS.IO network consists of 21 producing nodes elected by the EOSIO network community. The 21 nodes are connected in a full-mesh network, where each producer can readily communicate with all its peers.

Producing nodes are expected to run on a network server infrastructure that provides a robust, high performance, highly available set of producers, secured from nefarious access, and protected from arbitrary access that might negatively impact producer performance.

The key design goal for the core network is to keep the producing nodes focused on producing blocks and synchronizing among themselves. A producer may be deployed as a collection of worldly distributed servers equipped with high processing power and very large

volumes of memory, storage, connected together by high capacity redundant links, and protected by firewalls. Equally capable backup servers must account for the system integrity in case of server failure.

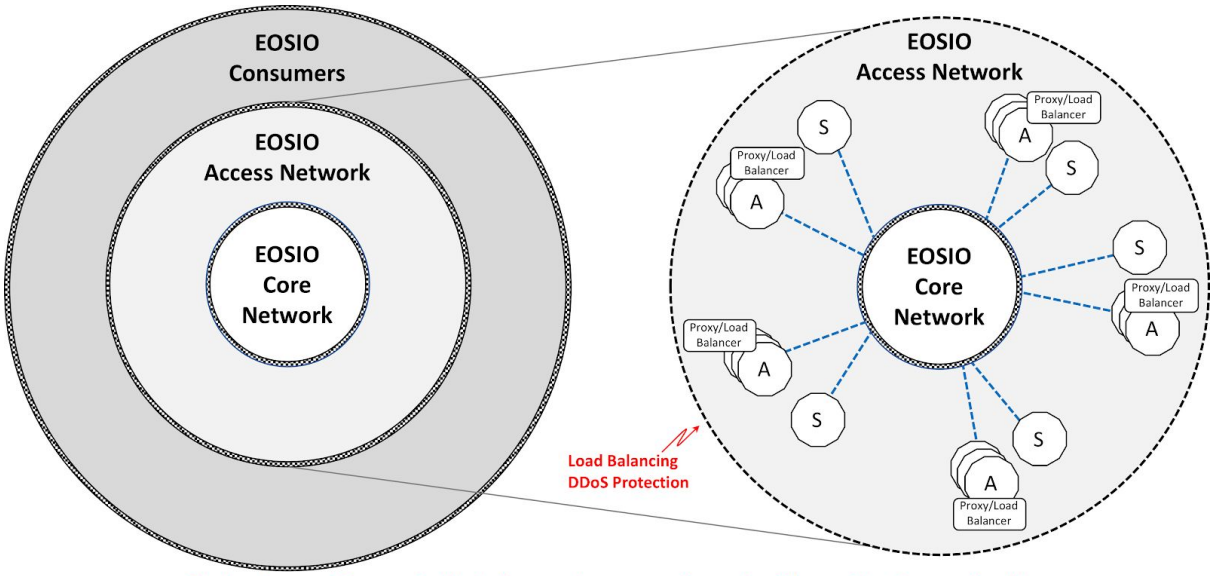


1.1.2. EOS.IO Access Networking

The EOS.IO access network aims to enable the core network to scale. It provides filtering and buffering to balance the load differences generated by the large number of block consumers and the small number of block producers in the EOS.IO core. The access network also establishes an environment for candidate block producers to establish their credibility and position themselves to be voted in as producers.

Nodes in the access layer are controlled and protected by entities willing and able to invest in the required infrastructure. Server nodes need to have significant processing, memory, and network capacities. The access network needs to be robust against attacks such as Distributed Denial of Service attacks.

While processing in core nodes is intensive and communication is heaviest among peers, access nodes must reduce traffic from a very large numbers of clients in the consumer network to only the essential to be managed in the core.



Access nodes can be designated as API nodes or Seed nodes, depending on the role to be performed by the specific node.

1.1.3. EOS.IO Consumer Network

The Consumer Network consists of the general users of the blockchain, interacting with it directly via cleos or indirectly using some application that interfaces with the blockchain.

Also included in the consumer network are Validating seed nodes that follow block production.

1.1.4. API Nodes

API nodes handle requests generated by cleos to execute transactions and query states. Communication between API nodes and cleos happens via http, while other nodeos nodes use the EOS.IO networking protocols. This offloads significant work from other nodes. A number of these nodes might operate collectively behind a proxy/load balancer.

API nodes perform an important role of "pre-processing" transactions using speculative state as they know it. This enables them to decrease the likelihood of faulty transactions making their way to producing nodes. Handling exceptions can significantly slow down a node's processing throughput. After processing action requests, the API node filters out the bad transactions and relay good transactions to one or more producer nodes.

Each producer node should have at least one associated API node.

1.1.5. Seed Nodes

Seed nodes communicate with other nodeos nodes to maintain synchronization with the producer nodes. Seed nodes may be producer candidates maintaining synchronization with

producers and servicing other nodes. Seed nodes might also provide access to other blockchains (inter-blockchain communication).

Seed nodes might also provide a layer of insulation between producers and other nodes on the network.

Validating seed nodes operate in the Access or Consumer Networks to track the validity of the blockchain. Seed nodes typically only communicate blocks (not transactions) using the EOS.IO network protocols, and are not configured to run the http protocols.

Each producer node must have at least one associated seed node.

1.1.6. Minimal Contracts

The bootstrap and operation of an EOS.IO Network involves many contracts: eosio.bios, eosio.token, eosio.msig, eosio.system, and proposals.

1.1.6.1. eosio.bios

Out of the box, nodeos starts up under the control of the eosio.bioscontract. This contract is used to set basic operating behavior, set account and global operating limits, set privileges, set producers, and establish authorization levels. By default, nodeos starts under the authority of the eosio account. The system is bootstrapped using the eosio account and remains under its control until such time that the eosio account relinquishes control to or shares control with another account or accounts using the eosio.bios contract.

1.1.6.2. eosio.token

This simple but powerful contract provides the fundamental currency management capability of EOSIO.

With this, the community creates its currencies, issues currency to its account holders, and transfers currency among its members.

1.1.6.3. eosio.msig

This contract enables multiple signatures at various required levels of authorization to be supported. This works in conjunction with the EOSIO flexible permission level capabilities.

1.1.6.4. eosio.system

This contract enables users to stake tokens, and then configure and vote on producers and worker

1.1.6.5. proposals.

The economics of the blockchain are effectively established and managed using this contract.

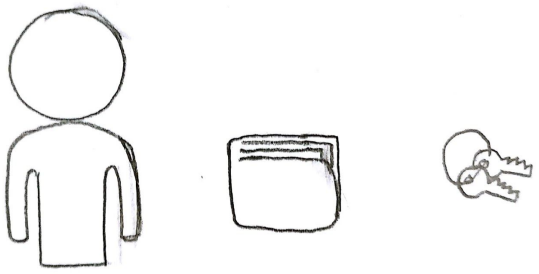
1.1.7. Plugins

- history_api_plugin
- history_plugin
- chain_api_plugin
- chain_plugin
- faucet_testnet_plugin
- http_plugin
- net_api_plugin
- producer_plugin
- txn_test_gen_plugin
- wallet_api_plugin

1.2. Blockchain Objects

1.2.1. Accounts, Wallets, and Keys

Each user is uniquely identified by their account. Wallets are used to store assets. Each wallet is uniquely tied to a key pair, where the public key represents the wallet's address, and the private key functions as proof of wallet ownership.



1.2.2. Authentication

Users gain access to their accounts though authentication. Each wallet is locked and unlocked with unique high entropy password that users keep as a secret.

1.2.3. Smart Contract

Smart contracts provide the business logic of Decentralized Apps. They allow users to perform actions on the network while signing to a Ricardian Contract that binds it to the DAC Constitution.

EOS.IO smart contracts provide an ABI interface. DApps trigger smart contract execution by providing JSON data structures via HTTP-RPC calls.

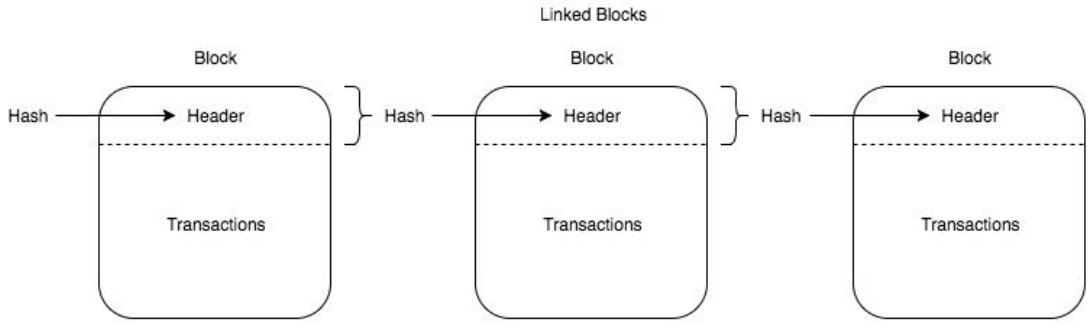
3.2.4. Actions and Transactions.

Actions are the behavior of smart contracts. They provide the high level functionality of smart contracts by sending and receiving payload data structures from the smart contract execution. They consist of JSON nested data structures with information about the atomic actions to be executed.

Transactions are collections of one or more atomic actions to be executed together. If one atomic action fails, the entire transaction is rolled back.

3.2.5. Block

Blocks are sets of independent transactions bundled together. Execution of smart contracts yields transactions to be achieved in a block.



3.3. Delegated Proof of Stake

Delegated Proof of Stake (DPoS) empowers tokens holders to select block producers through a continuous approval voting system. Anyone may choose to participate in block production and will be given an opportunity to produce blocks, provided they can persuade token holders to vote for them.

A holder of tokens on the DAC blockchain, who may not have an immediate need to consume all or part of the available bandwidth the tokens provide, can delegate or rent such unconsumed bandwidth to others.

EOSBake's Blockchain consensus is achieved via the DPoS consensus mechanism proposed by the EOS.IO platform.

3.3.1. Node

Node is a generic term that refers to any computing platform connected to the blockchain network infrastructure. Nodes might operate as block producers, front-end servers, back-end servers, amongst others.

EOSBake aims to provide hardware and software solutions for all kinds of EOS.IO Nodes (e.g.: BP, Full Node, Access Node).

3.3.2. Block Producer

The DAC governance mechanism recognizes that power originates from token holders who then delegate that power to the block producers. The governance process efficiently directs the existing influence of block producers to line up to the interests of the token holders.

Block producers are given limited and checked authority to freeze accounts, update defective applications, and propose hard forking changes to the underlying protocol. Before any change can be made to the blockchain the block producers must approve it. If the block producers refuse to make changes desired by the token holders then they can be voted out.

3.3.4. Proposals

In addition to electing block producers, token holders can elect a number of Worker Proposals designed to benefit the community. The winning proposals receive tokens of up to a configured percent of the token inflation minus those tokens that have been paid to block producers.

3.3.5. Incentive

The EOS.IO blockchain will award new tokens to a block producer every time a block is produced. A cap on producer awards can be set such that the total annual increase in token supply does not exceed 5%.

3.3.6. Staking

3.3.6.1. Resources

On the EOS.IO blockchain, there are three broad classes of resources that are consumed by applications:

- Bandwidth and Log Storage (Disk)
- Computation and Computational Backlog (CPU)
- State Storage (RAM)

3.3.6.2. Consumption

The EOS.IO platform allows each account to consume a percentage of the available capacity proportional to the amount of tokens held in a 3-day staking contract. For example, if an account on the EOS.IO blockchain holds 1% of the total tokens distributable pursuant to

that blockchain, then that account has the potential to utilize 1% of the state storage capacity.

While bandwidth and computation can be delegated, storage of application state require application developers to hold tokens, or stake them, until that state is deleted. If state is never deleted, then the tokens are effectively removed from circulation.

3.4. Side Chains

TODO

3.5. Network Infrastructure

3.5.1. Functional Requirements

In order to achieve mass adoption, blockchain infrastructures need to fulfill the following minimal functional requirements:

- 20k Transfers/s, based on Visa/MasterCard.
- 52k Social Interactions/s, based on Facebook.
- 100k Trades/s, based on BitShares.

3.5.2. Core Technologies

The semantics of a contract is defined with the standard general purpose programming language (C++) and compiled into the binary instruction format for stack-based virtual machines (WebAssembly). The EOS.IO platform provides a Operating System-like construct that allows the distributed execution of a Turing Complete Virtual Machine where contracts can be executed.

3.5.2.1. C++

C++ is the industry standard general-purpose programming language. It provides object-oriented, generic and imperative features, as well as low-level memory manipulation and good integration with hardware. C++`s design also has deep integration with UNIX based Operating Systems, which results in efficiency and system stability.

3.5.2.2. WebAssembly

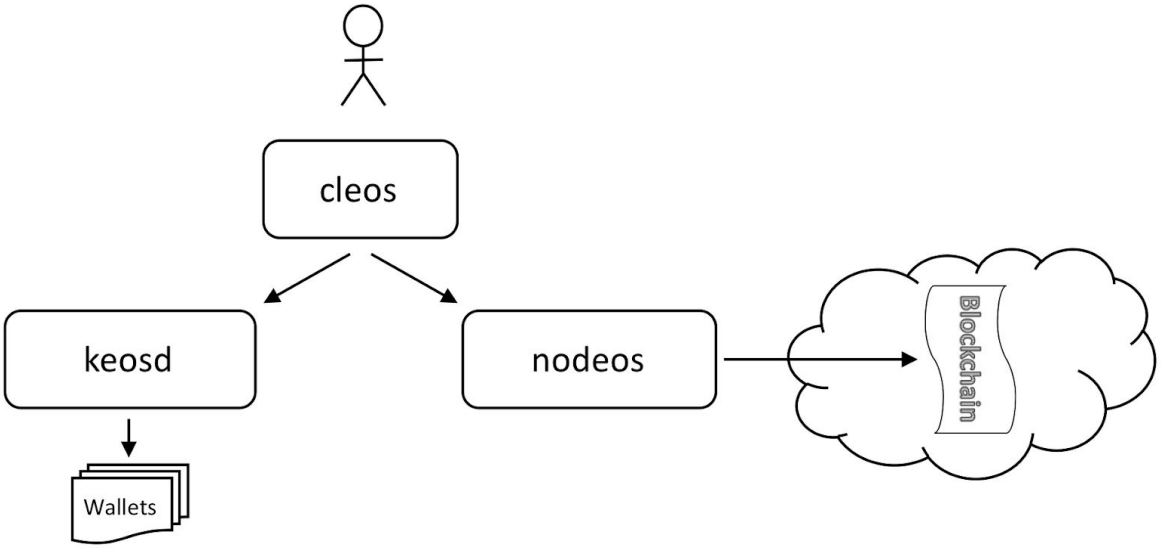
WebAssembly acts as a canonical bytecode format from which bytecode blobs are loaded into browser runtime. WebAssembly is the state-of-the-art technology endorsed by the World Wide Web Consortium (W3C), which means DACs are likely to achieve mass adoption due to the its standardized nature. WebAssembly`s fast execution enables seamless and responsive User Experiences.

3.5.2.3. EOS.IO Programs

Nodeos is the server-side blockchain node component, the core EOS.IO daemon that can be configured with plugins to run a node. It is responsible for running nodes and interacting with the EOS blockchain by creating blocks.

Cleos is the command line tool that interfaces with the User Interface exposed by nodeos as a REST API. In order to use cleos, the user needs to have the IP address and port number of a nodeos instance.

The program keosd is used to store private keys that cleos uses to sign transactions sent to the blockchain. Keosd runs on local machines and stores private keys locally.



4. EOSBake as an EOS.IO Infrastructure Platform

EOSBake's mission is to provide Free/Libre tools for flexible and customizable design of EOS.IO Blockchain Infrastructure.

4.1. EOS.IO Development Tools

4.1.1. Launcher

The launcher application simplifies the distribution of multiple nodeos nodes across a LAN or a wider network. It is configured via CLI to compose per-node configuration files, distribute these files securely amongst the peer hosts, and then start up the multiple instances of nodeos.

4.1.2. Eosiocpp

Eosiocpp is crucial for the development of smart-contracts. It is used to generate ABI files and can also generate helper functions that serialize/deserialize the types defined in the ABI for the contract development.

4.2. Yocto

The Yocto Project is an open source collaborative project coordinated by the Linux Foundation to help developers create custom Linux-based systems for embedded products, regardless of hardware architecture.

Yocto provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for all sorts of target hardware.

Yocto's main components are BitBake, OpenEmbedded, and Poky.

4.2.1. BitBake

BitBake is a python based make-like build tool specialized in cross compilation, although not limited to that. BitBake is inspired by Portage, Gentoo's package management system. BitBake uses recipes, which are metadata files with information about the packages to be built.

4.2.2. OpenEmbedded

OpenEmbedded is a cross-compilation and build-automation framework focused on creating Linux distributions for embedded devices, adopted by the Yocto Project in 2011. OpenEmbedded modularizes BitBake recipes in form of layers. Each layer is usually called *meta-layername*.

4.2.3. Poky

Poky is the reference distribution of the Yocto Project. It contains the OpenEmbedded Build System (BitBake and OpenEmbedded Core) as well as a set of metadata to get you started building your own distro.

4.3. Civil Infrastructure Platform

The Civil Infrastructure Platform (CIP) is a collaborative, open source project hosted by the Linux Foundation. The CIP project is focused on establishing an open source “base layer” of industrial grade software to enable the use and implementation of software building blocks in civil infrastructure projects. Currently, civil infrastructure systems are built from the ground up, with little reuse of existing software building blocks.

The CIP project intends to create reusable building blocks that meet the safety, reliability and other requirements of industrial and civil infrastructure. By establishing this ‘base layer’, CIP aims to:

- Speed up implementation of civil infrastructure systems;
- Build upon existing open source foundations and expertise without reinventing non-domain specific technology;
- Establish (de facto) standards by providing a base layer reference implementation;
- Contribute to and influence upstream projects regarding industrial needs;
- Motivate suppliers to actively support these platform / provide an implementation; and
- Promote long term stability and maintainability of the base layer of code.

With respect to project governance, a Governing Board is responsible for financial matters with respect to the project while the Technical Steering Committee oversees the technical direction of the project.

The GNU/Linux distribution CIP Deby complies to the cyber security standard for industry (IEC62443-4) to secure civil infrastructure from cyber attacks. ISA/IEC-62443 are a series of standards, technical reports, and related information that define procedures for implementing electronically secure Industrial Automation and Control Systems (IACS).

The Civil Infrastructure Platform (CIP) project aims to provide an open source base layer (OSBL) for embedded systems that require super long-term maintenance (SLTS). The OSBL consists of the super long-term support (SLTS) kernel, and the CIP Core packages. The CIP core packages are a set of industrial grade core open source software components and tools. The CIP Core project aims to implement reference file systems that use the CIP Core packages and can be tested on reference hardware.

4.4. OpenBMC

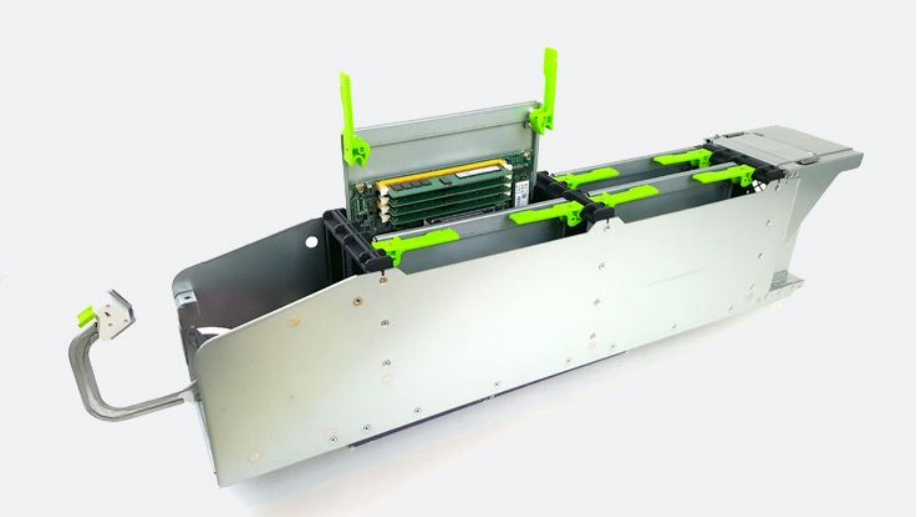
The OpenBMC project is a collaborative open-source project supported by the Linux Foundation and Facebook. It provides a Yocto based system management framework for the Baseboard Management Controller (BMC) software stack.

BMCs are controllers embedded in network infrastructure hardware. They are found as System-on-Chip (SoC) devices, which means they have their own memory, CPU, storage and IOs. OpenBMC is a GNU/Linux specifically tailored for the networking tasks to be managed by the BMC device.

The following devices are officially supported by OpenBMC, amongst others:

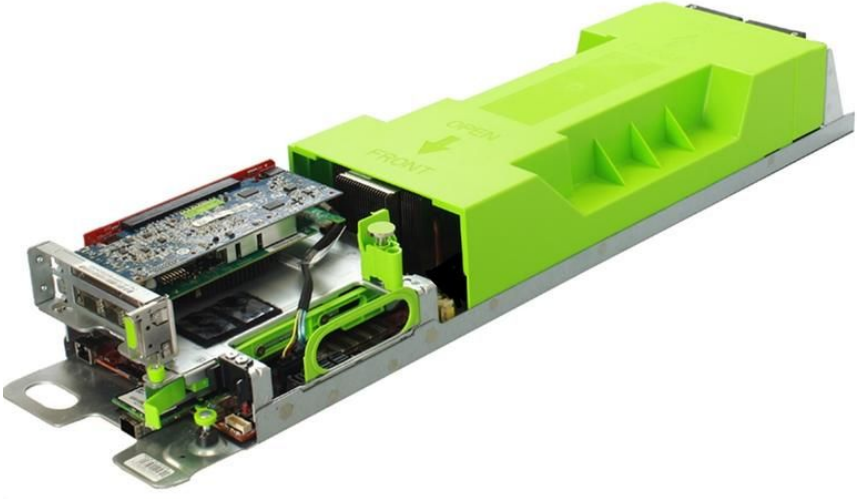
- Yosemite: an open source modular chassis for high-powered microservers. Yosemite hosts 12 dense computing power nodes to provide a complete front-end server system, network routing, wireless base stations, industrial Internet of Things, dynamic

web serving and more.



- Wedge 100: a Top-of-Rack cost optimized switch for web-scale data centers. Wedge 100 provides 10G/25G/40G/50G/100G server connectivity and 100Gb uplinks to the aggregation layer of the network.

- Tioga Pass: an OpenRack compatible dual sockets server Intel Motherboard. The Intel Motherboard V4.0 Tioga Pass is a derivative design of Intel's Cloud Reference board based on the Intel® Xeon® processor (Skylake) microarchitecture. The motherboard supports double sideboard Stock Keeping Unit (SKU) and single side sled designs in Open Rack Version 2 (ORv2).



EOSBake integrates the Yosemite, Wedge 100 and Tioga Pass devices in order to provide state of art transaction speeds for EOS.IO Networks.

4.5. Linux Package Management

GNU/Linux is the standard Operating System running most of modern network hardware infrastructure. It is quite literally the backbones of the Web. GNU/Linux applications are distributed as packages, which are fetched and installed to servers by the sysadmin through package managers.

Package repositories enable constant updates of applications for system administrators. Package management facilitates mass DApp adoption by taking advantage of a user interface that has already been largely validated inside GNU/Linux communities. Stakeholders are able to easily set up EOS.IO nodes.

Cloud computing providers sell server solutions based on popular GNU/Linux distributions such as Ubuntu, RHEL, CentOS and OpenSUSE. Appart from OpenBMC, a few other GNU/Linux projects that focus on network hardware:

- Open Network Linux
- OpenWRT
- OpenvSwitch

Because Yocto provides package management support, DAC stakeholders are free to choose the OS layer in accord to their needs for their nodes. The following package formats

are widely adopted by popular GNU/Linux distributions, and are supported by Yocto and EOSBake:

- **.deb**, supported by Debian, Ubuntu
- **.rpm**, supported by RHEL, CentOS, OpenSUSE
- **.ipk**, supported by Ångstrom, OpenWrt

4.6. Quality Assurance

Embedded software is increasingly used within industrial automation, as processes become more complex and vendors differentiate commodity devices and improve safety. Software failure might result in damages on brand reputation and even loss of life.

EOSBake relies on QA-SYSTEM's QA-C and QA-C++ tools to perform CERT and IEC 61508 Quality Assurance of DApp source code and smart contracts.

The IEC61508 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems), is an international standard widely used in Industrial Automation.

CERT is maintained by the Software Engineering Institute (SEI), a research and development center primarily funded by the U.S. Department of Defense and the Department of Homeland Security. The CERT Division at SEI is operated by Carnegie Mellon University and responsible for publishing these standards. The CERT® Secure Coding Standards for C and C++ are standards that provide rules and recommendations that target insecure coding practices and undefined behaviors that can lead to exploitable vulnerabilities.

5. OpenEmbedded Layers

Governance requires auditability of the full software stack of the blockchain network infrastructure. OpenEmbedded Layers provide application to hardware (BSP) level integration by providing metadata instructions that automate the code production chain. Source code and recipes are available in public repositories, enabling easy auditability and establishing trust for the network infrastructure.

5.1. OpenBMC

OpenBMC's Yocto support is provided by OpenEmbedded Layers maintained by Facebook, which means there's good support and an active community. The main layers available at the official repository are:

- **OpenBMC Common Layer**: common packages and recipes to be used in different types of BMC.
- **BMC System-on-Chip (SoC) Layer**: SoC specific drivers and tools. Includes the bootloader (u-boot) and the Linux kernel.

- **Board Specific Layer**: board specific drivers, configurations, and tools.

This enables full stack integration between BMC hardware and blockchain applications such as full nodes, block producers, and access nodes. Coordination between high and low levels of abstraction is achieved at build time with Yocto recipes, which results in large throughput (high TPS) at runtime.

5.2. meta-debian

Deby is the Poky-based reference implementation of the CIP Core project. Deby is a reference distribution built with poky and meta-debian, a layer for the poky build system that allows cross-building file system images from Debian stable source packages.

CIP Deby's repository contains a few OpenEmbedded Layers. EOSBake's meta-eosio expands upon the functionality meta-cip-common and meta-cip-qemux86-64 layers.

5.3. meta-eosio

The **meta-eosio** layer provides support for the basic EOS.IO infrastructure. Recipes are written so the source code is fetched from trusted official EOS.IO repositories, and binaries for applications such as cleos, keosd, and nodeos are built for the target architecture. Development tools such as launcher and eosiocpp are also supported. Coding templates for contracts, plugins and libraries leverage EOS.IO platform versatility, enabling flexible node design.

5.4. meta-eosbake

Recipes for EOSBake GNU/Linux Distribution based on CIP Deby and OpenBMC are provided. OpenBMC BSP layers are integrated into meta-eosio, providing support for the Yosemite, Wedge 100, and Tioga Pass platforms.

Basic example recipes for EOS.IO Nodes are provided, and users are encouraged to use them as boilerplate reference for their own projects. Flexibility is one of StraberryDAC's goals, so many different node designs can be implemented by developers within the Yocto workflow. Recipes are organized in terms of EOS.IO network architecture, allowing developers to build and deploy highly customized nodes for different scenarios and applications.

6. R&D

Some social aspects of DACs can only be learned from careful observation of real world experiences.

EOSBake R&D documents cases of DAC implementations around the world for careful community analysis. This allows for community learning by closing the knowledge feedback loop, thus allowing for constant improvement and stability of governance mechanisms.

6.1. FollowMyVote

Follow My Vote develops open source end-to-end verifiable blockchain voting software for use in government-sponsored elections worldwide. EOSBake's vision is to team up with FollowMyVote to provide the hardware+software solution for one election as an experiment.

6.2. TundraX

TundraX is GCH's EOS.IO based Social DEX. EOSBake's vision is to provide the hardware+software solution for the deployment of FROST nodes around the world.

7. Team

The EOSBake team is composed by the following members:

- Sky Trotter: GCH
- Bernardo Rodrigues: Systems Architect
- Leslie Aker: Cryptographer and Advisor
- Otavio Salvador: GNU/Linux Expert and Advisor
- Syed Jafri: EOS Community Advisor

8. Partners

The EOSBake development will be carried by project partners O.S. Systems and BitSystems. Both partners will be hired by Global Crypto Holdings Inc. under a software development contractual agreement.

- **O.S. Systems** is specialized in the development and customization of embedded operating systems and BSPs based on Linux. They provide software development and consultancy services specialized in the Yocto Project. O.S. Systems is based off of Pelotas, Brazil.

- **BitSystems Consulting** is an end-to-end blockchain product strategy, design, and development company dedicated to bringing blockchain solutions and insights to the world. BitSystems is based off of Cambridge, Massachusetts.

9. Development Roadmap

The EOSBake's development roadmap can be divided in three phases: a) meta-eosio, b) meta-eosbake, and c) Quality Assurance (QA).

TODO redo this

10. Business Roadmap

EOSBake’s business map consists of the following phases:

- **Jul-Aug 2018**: white paper, website, pitch-deck, business plan, scaling plan
- **Sep-Dec 2018**: series A funding, development phase 0 (MVP)
- **Q1 2019**: alpha/beta testing, testnet launch, development phase 1
- **Q2 2019**: penetration testing, bug bounty hackathons, development phase 2
- **Q3 2019**: marketing, scaling, R&D

11. Budget

The following table summarizes the estimated costs necessary to make EOSBake viable. Every item includes the cost of a specialized consultancy working together with GCH on the task execution.

TODO