www.chainfrog.com

chainfrog

# ISSUSE AND FEATURES TO CONSIDER WHEN SELECTING A BLOCKCHAIN SYSTEM

**Find us at www.chainfrog.com**

# Licenses

- Most blockchains are open-source

  (and you should not select a closed source one)

- If you are going to change a blockchain codebase the license it carries is important

- Generally the license for a given project is selected from one of three

- Check with your lawyer for details (we’re blockchain experts, not legal experts)

# 1) MIT License

- Very permissive open software license:
  - Do what you want with the code; ship only binaries, rewrite part, rebrand, repackage, safely add your own code
  - Makes no reference to IP

- Limitations and obligations:
  - Damage waiver (you can’t sue if their code causes you a problem)
  - Add a copy of the license
  - Add the copyright notice

# 2) Apache 2.0 License

- Very permissive open software license:
  - Do what you want with the code
  - Release your modifications under a different license if you like
  - No obligation to release source
  - Explicitly grants right to use patented software under the license terms
- Limitations and obligations
  - Damage waiver (you can’t sue if the code causes you a problem)
  - Add a copy of the license to the unmodified parts

# 3) GPL3 License

- “Free software”  license:
  - Do what you want with the code
  - You can keep your work private and secret
  - “Copyleft”

- Limitations and obligations
  - Damage waiver (you can’t sue if the code causes you a problem)
  - You must give anyone who you distribute binaries to a copy of the source code
  - The source code must contains the GPL3 license
  - So anything you add, merge, or improve becomes covered by the GPL3 license
  - You lose IP and control; it’s “out there” once released

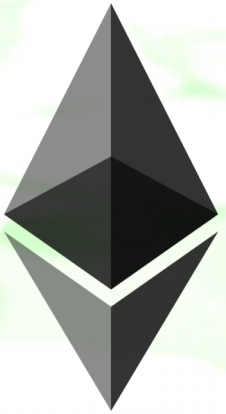
# Other Issues

- Cost (do you need to buy native tokens, pay a license fee, buy dedicated hardware, cloud computing)
- Consensus protocol (can also be a cost)
- Transaction speed (how quickly can you push data onto the system)
- Transaction size limitations
- Programming language used
- Focus and purpose of the blockchain
- Where are you going to run it? Cloud, private servers, mix of both? Windows, Linux?

# Bitcoin Fork

- Covered by MIT license
- Well established highly tested code base written mainly in C++
- 80 bytes/transaction
- 10 transactions/second
- Would require a lot of coding to transform into something more useful (but see later systems)

# Ethereum



- Different components have different licenses:
  - Some GPL3, some MIT
- Different version coded in Go, C++, Java, Python, Ruby
- Development funded by crowdfunding in 2014
- Launched in 2015
- A public blockchain with an associated cryptocurrency called Ether (ETH), and a more complicated (Turing complete) scripting language called Solidity, which is like Javascript
- 100 tx/s
- Can deploy private “testnet” version, but privacy is not a core concern
- Expensive to push lots of data onto the public version
- Lot of infrastructure available (compilers, monitors, explorers)
- Focused on “smart contracts”

MONAX

# Eris

- A fork of Ethereum
- Removed Nakamoto consensus from Ethereum, and replaces it with Tendermint, a Proof of Stake consensus system
- Adds better private and permissioned blockchain capabilities
- Eris “smart contracts” also should run on Ethereum
- Licensed under GPL3
- If it’s the same as Ethereum, transactions can be up to about 800kB
- Scalability is a bit complicated in Ethereum and Eris though, as there’s the “gas limit” that restricts transaction size and throughput.

# MultiChain

MultiChain

- Currently only binaries available; CEO of company confirms GPL3 and paid licenses will be available
- Forked from Bitcoin, but with a “round robin” signing consensus protocol, meant for permissioned blockchains
- 8 Mb/transaction
- 100 transactions/second
- Supports “private data streams”; encrypted data stores only readable by authorised parties. Data is in JSON format
- Supports multiple “native assets” - own cryptocurrencies.

# bigchainDB

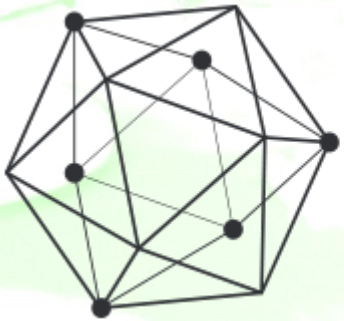
- Focused on scalability; built on rethinkDB noSQL database
- Claims to support “a million transactions per second” and large data volumes
- License is either AGPL3 (a more copyleft version of GPL3) or you can negotiate a commercial license for a fee
- However, the company that makes rethinkDB shut down in October. Presumably bigchainDB is looking for a new supplier as we speak.
- Consensus is through multisig signing of transactions. There are no blocks…
- Has “big data” query capability

# Corda


- Built from scratch by R3 consortium, funded mainly by banks
- Focuses on finance transactions, privacy, permissions, workflow
- A selection of consensus protocols available
- Recently open sourced with Apache 2.0 license
- Written in Kotlin (a new Java-like programming language)
- Supports “distributed apps”, which are basically “smart contracts”

# Hyperledger

- This is the “blockchain” project that the big companies are supporting.
  - Coordinated by the Linux Foundation
  - Three main projects are:
    - Fabric (IBM blockchain for smart contracts)
    - Sawtooth Lake (Intel blockchain using modular components)
    - Iroha (Soramitsu blockchain focussing on moble apps, new)
- Yes, there are three different blockchain systems...



HYPERLEDGER

# Fabric

- Tutorial on how to set up a development environment at http://www.chainfrog.com/ibm-hyperledger-fabric/
- It uses:
  - Docker, a software containerization platform, to ensure that if you run Fabric on a native platform, all the required packages and libraries are standardized. A Node.JS npm provides access to a Fabric SDK.
  - Or, a Virtual Machine running on VirtualBox, and Vagrant to ensure the VM is Ubuntu configured to ensure a standard development environment.
  - Smart contract focussed, using Go language contracts
  - Practical Byzantine Fault Tolerance, or Sieve, as a consensus protocol.
  - Designed as a permissioned blockchain
  - Best deployed on IBM bluemix clound platform

# Sawtooth Lake

- Also uses a VM and Vagrant to provide a standardized environment for execution and development
  - Core “sawtooth lake” code provides peers/validators
  - “MarketPlace” JSON data structures and objects provide the means to create and execute transactions on the blockchain
  - Uses a consensus protocol that really requires trusted execution environment hardware and software (i.e. Intel SGX)
- Tutorial available at https://intelledger.github.io/tutorial.html