Lamden: a framework for the rapid development and deployment of blockchain technologies.

Stuart Farmer*, Mario Hernandez†, and James Munsch‡

The Lamden Project, White Paper version 0.2.1

January 5, 2018

Abstract

Blockchain application development is unnecessarily hard, partially because of how new the technology is, but also because of certain imposed limitations on the current status quo of how blockchains are used today. In this paper, we discuss the problems with current blockchain implementations and a solution to these problems. We introduce Lamden: a software package of development tools that includes a blockchain generator, community collaboration tools, and independent chain-to-chain communication to create vast networks of blockchain applications that can transfer any asset across them.

Keywords: decentralization, cryptocurrency, distributed systems.

*stu@lamden.io
†mario@lamden.io
‡james@lamden.io

1. Motivation

In the past several years, we have seen an influx of blockchain technologies hitting the market. These technologies are touted as a means to an end of the centralized banking and financial systems. However, these blockchains rely on flawed systems that do not take into account the application of said technologies into the current infrastructure space. As a result, blockchain technology is segregated from the mainstream financial systems, and thus is ironically hindered by achieving its end goal.

The Lamden Project is an effort to create software development tools that increase the mass adoption of blockchain technology so infrastructure can revolutionize the market systems and cause realistic innovation within the markets.

Laid out in this white paper are our critiques of the current state of the markets, and how we plan on improving this space.

2. Problems with Current Blockchain Implementations

There are many fundamental flaws with the blockchain architecture that prevent it from becoming a mainstream technology. These flaws exist on both the user's side as well as the developer / service provider's side. Laid out are the areas in which we improve upon the current blockchain implementations with this project.

Transaction Fees

Firstly, let's take a look at the problem with transaction fees from an end user standpoint. The key to success in the crypto space is mass adoption of Bitcoin in the international market. As it stands at this time of writing, the average transaction fee for a Bitcoin transaction is at around 6%[1]. Compared to centralized systems such as PayPal[2], Mastercard[3], and Visa[4], Bitcoin is actually about twice as expensive.

It should be noted the fundamentals of cryptocurrency are why users feel justified with this discrepancy. The decentralized nature of ownership outweighs the additional costs. However, on a macroeconomic scale, cost efficiency will outweigh fundamental allure.

"But what about Ethereum? Their transaction fees are only 0.3%!"

While this is true, the Ethereum blockchain was not designed as a transfer of value network[5] like Bitcoin was, but rather a smart contracts network to enable trustless and decentralized software systems. We have to remember that users instead pay for the number of operations that occur on the chain and that this number of operations continues to increase as more and more complicated smart contracts are deployed. A major selling point of Ethereum was the

ability to add smart contracts as 'libraries' that function in the same way as frameworks in other software development stacks which is to automate operations and processes so users do not have to rewrite them over and over again. Thus, we can safely assume that over time, as more complex smart contracts are made and more libraries are integrated, the cost of interacting with the main Ethereum blockchain will rise. This mechanism by charging developers on operations disincentivizes innovation and complex smart contract systems that Ethereum needs to be a long term success.

Service Name	**Average Transaction Cost**	**Paying Party**
Bitcoin[1]	~6% (variable)	Sender
PayPal[2]	2.9% + $0.30	Receiver
Visa[4]	Based on merchant type. As low as 0.20% for utility payment, to as high as 1.50% for standard merchants.	Receiver
Mastercard[3]	Based on merchant type. As low as 0.00% + $0.65 for utility payment, to as high as 2.95% + $0.10 for standard merchants.	Receiver
Stripe[6]	2.9% + $0.30	Receiver

Table 1: Various payment services and their transaction costs.

Bloated Chains

Carrying on with the Ethereum model, because the ledger is public, anyone can deploy whatever smart contracts they'd like. As a result, even though the blockchain is only two years old, the size of a full node is 242GB[7], which is even larger than the current size of a Bitcoin full node which is around 152GB[8]. Having to install a full node, or at least a light chain which is still around 20Gb of data, is a hefty barrier of entry for someone who just wants to dabble in smart contract development.

Compare the time it takes to fully sync a node with the time it takes to install Python, Node.js, or any other programming environment to start poking around. The inconvenience and download size alone are unfeasible to most developers and companies, let alone the average user who is curious about blockchain technology.

Transactions per Second (Scalability)

Modern blockchains have a notoriously low throughput rate which causes transaction times to be slow, and a backlog of pending transactions to pile

up. This is an unacceptable replacement for the current financial systems that process millions of transactions per day.

For example, Bitcoin has a throughput rate of 3 transactions per second. Likewise, Ethereum boasts a mere 15 transactions per second[9]. In comparison, PayPal released a case study in May of 2015 where they were able to create a system throughput rate of over a billion transactions a day. That translates to about 11,600 transactions per second[10].

This disincentivizes enterprises that deal with assets in a traditional high throughput database structure to sacrifice capacity for the benefits of high record maintenance.

Mining and Resource Efficiency

Another barrier towards mass adoption is the arbitrary hashing functions that mining rigs perform to 'prove work.' These machines take up massive amounts of resources in energy costs that make no sense to the standard enterprise user. Whereas this is an approach to distributing tokens and coins on a main chain, it does not translate to the business setting where capital resources are optimized. Inefficient processing makes no sense to the modern business, and thus these types of pointless hashing functions are impeding widespread adoption.

Integration and Adoption

Blockchain software is very new and the human capital resources are scarce. It requires a very highly technical and specialized individual to consult and develop for blockchain applications. Compared to other database types in a corporate setting, setting up a blockchain is time consuming and expensive.

Whereas most modern database systems follow a similar development and deployment paradigm, blockchains are so fundamentally different that there is currently not a solution in place to transforming the development process into something that resembles other database systems on the market.

Furthermore, development environments, package managers, testing suites, continuous integration environment, etc. which exist for nearly every other tech stack in existence do not exist for blockchain software. This prevents organizations from integrating this technology into their current workflow.

Migration, Upgrades, and Future Proofing

For many interested in blockchain technologies for their enterprises or businesses, the lack of ability to migrate data to a new chain, or to upgrade at a time that is best for a business is a pitfall. Those interested in mainchain and public ledger systems are reliant on foundations to release new code or the decentralised community where there is little authority or certainty to the outcome. Furthermore, trying out different systems to see what works best for an organization is cumbersome.

3. The Solution

The Lamden Project proposes the creation of a suite of development tools that allows rapid development and deployment of private blockchain systems that mimics the modern development process featured in such tech stacks as Node.js or Python which provide a plethora of easy to use tools, extensive and robust documentation, and vastly popular support communities.

The project is broken up into three sections, Saffron, Flora, and Clove, which each add a layer towards the end goal of providing hyperfast blockchains for developer to test, experiment, and deploy across a network of other blockchain systems.

Saffron is the generator tool to deploy private chains on an internal network without any hassle. Flora is a central repository for smart contract packages and templates, private chain naming services, and blockchain discovery. Clove is the network that handles payment channel swap processes between chains and facilitates communication between blockchain apps on the network in a way that remains trustless and decentralized.

Saffron: A Generator Tool

Instead of having a single mainnet which suffers from the problems mentioned above, breaking up blockchains into individual use cases and having them interact with each other when needed solves many problems with low blockchain adoption rate. For example, an organization would have their own blockchain for their own web app. This allows them to have complete control over the technology that they use as well as dedicating their own computing resources to running that blockchain, rather than wasting effort on a main chain. Furthermore, 'gas' in the case of an Ethereum chain can be eliminated and the difficulty of a network can be turned down to a more reasonable level, thus enabling free transactions across their application. This makes a blockchain act more like a web server than a traditional chain and can then be used to replace and upgrade existing database structures internally.

Motivation for Independent Consensus

What makes blockchain technology work is the fact that miners are independent actors that verify transactions that occur so it is near impossible to inject false information into the chain. However, in traditional settings, miners are rewarded in cryptocurrency from transaction fees. Because transactions are free on the Lamden protocol, a different rewarding mechanism must be instilled so that miners are still motivated to provide independent verification of transactions[11].

It is possible to create proof-of-work, proof-of-stake, and any other rewarding schemes by using smart contracts. In this way, a smart contract can be called upon to calculate the reward and recipient, and reward any type of asset on the chain. Instead of just the native currency of the chain (which is initially Ether), ERC20 tokens or novel cryptoasset implementations can be rewarded as well,

providing even more customization to private chains. And because transactions and gas is free on the networks, calling a contract once a block, for example, is not expensive.

```
pragma solidity ^0.4.15;

contract RewardedAsset {
    string public constant symbol = "AU";
    string public constant name = "Digital Gold";
    uint8 public constant decimals = 0;
    uint256 _totalSupply = 1000000000;
    uint8 public reward = 10;
    uint256 lastBlock;

    address public owner;
    mapping(address => uint256) balances;
    mapping(address => mapping (address => uint256)) allowed;

    function disperseRewards (bool success) onlyOwner() {
        if (lastBlock < block.number) {
            _totalSupply += reward;
            balances[block.coinbase] += reward;
            lastBlock = block.number;
        }
    }
```

Listing 1: Example of a simple proof-of-work contract that could be deployed on a chain to simulate classical proof-of-work.

Generator Tool

Saffron is taking design queues from other package management CLI (Command-Line Interface) tools such as `pip`[12], `yeoman`[13], and `npm`[14] to generate private blockchains as their own development projects. Generation through the CLI tools is as simple as '`saffron init`'. Through this CLI application, you will be able to interact with the chain in a simplified API that abstracts most of the complexities of blockchain technologies from you.

The initial blockchain technology that we are implementing is Ethereum, and so developers will be able to initialize these chains, create accounts, mine blocks, and deploy contracts in a sub-processed instance of `geth`, Ethereum's client written in Go.

By abstracting the main interaction components away from `geth` into a CLI tool, we are able to then provide a RESTful API service for interacting with these private chains. A RESTful API allows for blockchains to be interacted with across the Internet in any programming language interface. If a programming language supports HTTP requests, they will be able to interact with Lamden generated private chains. This allows integration into any tech stack, and allows us to create drop in libraries for many modern language frameworks such as Swift

for iOS development, Java for Android development, and even embedded C for Internet of Things applications.

Furthermore, Saffron will provide a local server with an Ethereum blockchain explorer that can be public facing so that the wider community can monitor and see the activity occurring on the chain.

A web service to manage the chain from a graphical web interface similar to Parse will also be included out of the box. Finally, the stack will incorporate optionally deployable instances of IPFS / IPNS and Tor.

Deployment

After the development of a private chain, an organization will want to package and deploy that in a way that can integrate into their internal technical stacks. To do this, Saffron will export Docker files for quick deployment.

Architecture

Beyond creating a private chain instance, Saffron contains the necessary methods needed to accept payment channel transactions and interact with the larger Lamden chain. Thus, Saffron should be thought of as a type of interface. Each Lamden Wrapper instance must adequately conform to a set of monitoring and interaction protocols to guarantee that the chain is trustless and that transactions can be independently confirmed.

If Saffron can achieve these functionalities, then the internal blockchain driver does not have to be based off of Ethereum. Into the future, Saffron will support arbitrary blockchain technologies such as Bitcoin, Litecoin, Zcash, Monero, Neo, Tezos, Eos, and more as long as they can support payment channels in a trustless fashion. Even novel blockchain technologies that adhere to the wrapper protocol will be supported, creating a completely custom blockchain-to-blockchain interfacing network.

The Wrapper will ultimately support any blockchain technology as long as it adheres to certain specifications so that it can interact with Clove. This future-proofs the Lamden system so that no matter what technologies exist in the future, there will always be a network to interact with them. This also solves the problem of blockchain migration, as businesses, enterprises, and applications can upgrade to the latest blockchain projects and technology as they come out to keep ahead of the competition.

Flora: Decentralized Package Manager and Public Repositories

Apart from simple blockchain generation, the main driver of technology adoption is not the base use cases of the core tech, but rather the surrounding communities and feature-sets built by them. You can see this with the Ethereum community, and more with newer smart contract systems that are starting to spur innovation. Their cryptocurrency mechanisms are not what sets them apart from Bitcoin or Litecoin. Instead it is the development community surrounding and developing smart contract applications that affords wider adoption[15].

Similarly, package repositories that allow users to pull in packages from the wider community, comment on them, and even develop for them solidifies a base of intellectual capital that leads to the success of the project and greater innovation in the field.

The purpose of Flora is to host a decentralized package repository, namespace system, and discovery tool so that developers of Lamden chains can connect with other developers, play off each other's innovations, and lead to a greater state of cryptocurrency.

Public Package Manager

```
pragma solidity ^{{solidity_version}};

contract {{contract_name}} {
    string public constant symbol = "{{symbol}}";
    string public constant name = "{{asset_name}}";
    uint8 public constant decimals = 18;
    uint256 _totalSupply = {{total_supply}};

    address public owner;
    mapping(address => uint256) balances;
    mapping(address => mapping (address => uint256)) allowed;


...


{
    'solidity_version':'0.4.15',
    'contract_name':'Testcoin',
    'symbol':'TST',
    'asset_name':'Testcoin',
    'total_supply':'1000000'
}
```

Listing 2: Example of a Solidity Template for ERC20 tokens with corresponding JSON input. Installation of such contract would be accomplished via '`flora install erc20 -args`', or from Saffron administration panel.

Like other development suites, Lamden has the ability to create smart contract packages. Using a macro templating system, we are able to abstract common Solidity contract types such as ERC20 tokens, Ethereum Name Services, and the Etherdelta decentralized exchange, package them up into contracts that only take a set of arguments, and allow deployment across private chains in as simple as a manner of running '`hpm install ens`' from a CLI console, or sending an API request to do the same through the administrative dashboard included in Saffron. This takes the headaches out of redeploying commonly used smart contracts, and allows the packaging of highly complex smart applications for quick distribution.

As more blockchains are added to the Wrapper, support for their contract languages will be added to the Hub as well. Developers can upload their own contracts to support the community. Interaction with the Hub will be over the Interplanetary File System (IPFS) so that all services remain distributed and decentralized.

Name Directory Service

Chains will have to communicate with one another to execute payment channels. To do this, a naming service must be implemented. This naming service assigns a common name and a key pair to each chain so that other chains can discover them and sign transaction requests to them.

Furthermore, users will be able to search and discover new blockchain applications. These are private chains developed with Saffron that have connected to Clove. Each of these chains will be able to register for a name and fill out a profile about their service which can include links to marketing sites, connection information for miners, and anything else that would lead users to more valuable information about their service. That name will be used to locate and send information to through Clove for payment transactions, interaction with the chains, and the general use of the native Lamden token.

We have also been exploring options in using the Token for other developers to be able to register their modules to which usage metrics could generate token liquidity between members of the developer community.

Clove: Anonymous Router Enabling Chain-To-Chain Interfacing

The main technological feat of the Lamden Project is its cryptographic ledger and routing system that connects private chains together. By instead of recreating an entirely new chain and instead shifting focus towards distributed private chains only tied together by a routing infrastructure, we can avoid blockchain bloat, the need to 'sync' with the network before interacting with it, increasing transaction costs, and guarantee a future-proof platform that will last years regardless of the further improvements upon general blockchain technology.

Routing

Clove is primarily the infrastructure that links private chains together over a common protocol, much like a telephone network. The Ledger takes care of interaction between all chains and provides a common set of interface options to that chains can be treated the same ways in computer programming environments and platforms for further integration away from Flora.

A main value add of Clove is the absence of transaction fees. In exchange for the benefits of connection to Flora and exposing a private chain to the wider community which increases public adoption, a private chain dedicates a certain amount of computing power to the ledger overall. Because the ledger is absent of difficult hashing functions, and serves merely as an intermediary to process packets of information between chains. The computing load to connect

and serve information is comparable to a modern day web server rather than a standard blockchain miner[16]. Blockchain application providers expose a single public entry point onto the IPFS and distributed computing cluster for incoming transactions, and connect to the public Ledger node for outgoing transactions.

Payment Channels for Cross-Chain Transactions

While retaining a trustless and decentralized system, it is required for a system of cross-chain payments to exist so that blockchain applications can interact with each other freely. This is the main draw of Clove. To do so, the Ledger implements Lightning Network[17] inspired payment channels to provide a secure and decentralized method of exchanging funds across blockchains. Transactions initiate from private chains, and utilize the Ledger as a public service for managing secure and guaranteed remittance of funds from chain to chain.

The Ledger is a public service that distributes the work of smart contract deployment and identity verification for chain to chain transfers between participants connected to the main network. Private chains are free to implement their own methods by which to transfer funds to each other, but the public Ledger is verifiable by consensus of participants on the wider network[18], thus providing a stronger sense of trust to cross chain transactions.

Payment channels are represented in pure Solidity smart contract code that run on each private blockchain. When a user wants to initiate a swap, they send a request both Clove service and the chain they want to swap with outlining the terms of their expected transfer. The request is comprised of the sender and receiver address on both chains, as well as the value of the asset being swapped. For example, a payload would look like this:

```
{
    Private-chain-1 : {
        Sender-address : 0x000,
        Receiver-address: 0x000,
        Amount: 100,
        Token-contract-address (optional): 0x000
    }
    Private-chain-2 : {
        Sender-address : 0x000,
        Amount: 1000,
        Token-contract-address (optional): 0x000
    }
}
```

Listing 3: Example of the payload from a payment channel.

When the computing cluster receives this payload, the cluster will compile the Solidity contract code into bytecode that must be deployed onto each chain within the next block. The opposing chains' bytecodes are sent to each other and signed with a public identifier key so that they can independently verify

that the correct contract code was indeed deployed in the next block on the chain.

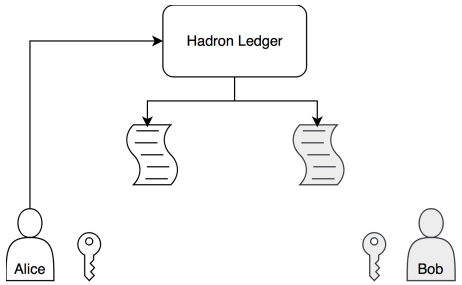

Figure 1: Alice initiates a swap by sending a message to the Ledger. The Ledger then takes the swap requirements and compiles them into Solidity bytecode to be deployed on each chain.

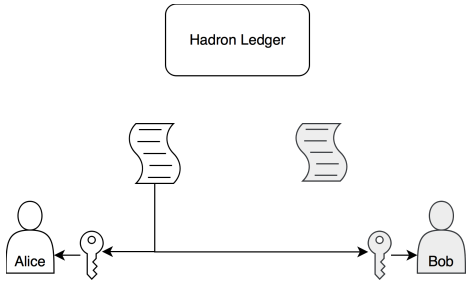

Figure 2: The bytecode is signed with each chain's public key and sent to both parties. Alice deploys their contract. Bob keeps a copy to verify later.

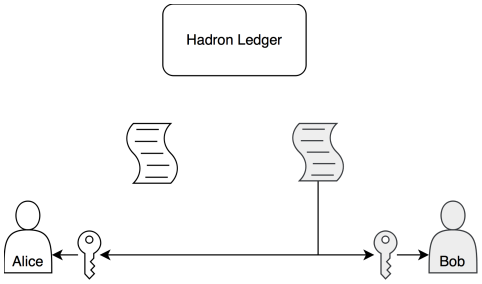

Figure 3: The same occurs for Bob.

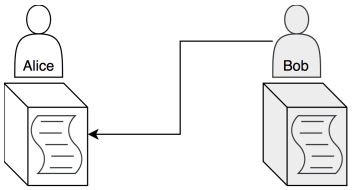

Figure 4: Assuming that this is an agreeable swap, both chains deploy the contract, and check to see if the other party has done the same by searching for the bytecode on the latest block. If this has not occurred, the chain does not have to follow through with any obligations, and the transaction reverts without loss of funds.

If the originator chain fails to deploy their contract while the recipient chain follows through, the contract on the recipient chain will self-destruct and remit funds back to their account. On the flip side, if the originator sets up a contract, but the recipient does not agree to the terms, the originators contract will be destroyed they will receive their locked up funds.

Assuming that both parties are interested pursuing the swap, they must then follow through with the agreed upon obligations on each others' chains, which executes the payment channel regularly [19].

To make sure that the payments have been sent through in a timely fashion, each chain can 'ping' their contracts which invokes a time-lock method [20]. If the contract is too many blocks behind the agreed upon time-lock, it is assumed that the other party is not going to finalize the payment channel, and the contract is destroyed. This incentives the opposing party to follow through with their obligations because they risk losing their funds. It also incentives the originator to ping their contract as soon as possible, because in the situation where the originator has received funds on the opposing chain but not on their chain, they can remit their funds back to themselves and in effect have gained 'free' assets on the opposing chain because the time-lock had expired on their own to claim the assets.

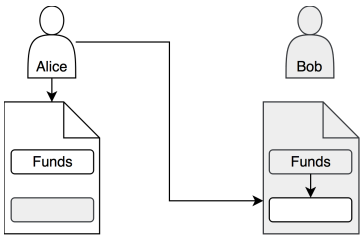

Figure 5: Alice's best economic option is to submit the funds and resolve their own contract as soon as possible, because if Bob acts before them, Alice will lose funds. Inversely, Alice could also profit if Bob decided not to redeem their funds. So it is also in Alice's best interest to resolve their contract as soon as possible

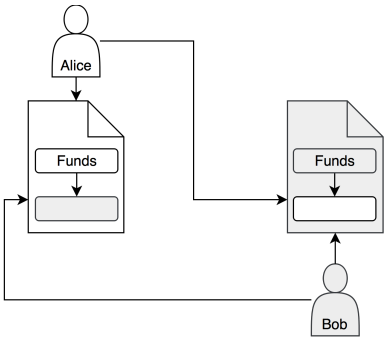

Figure 6: The best case scenario for both parties, therefore, is to resolve the payment channel on the opposing chain, and resolve the contract on their own chain as soon as possible. By adding economic incentive to do this, the network benefits from quick chain-to-chain transfers.

4. Example Workflow

Firstly, you will install Saffron tool designed for developing blockchains. You will run '`saffron init`' inside of a project directory and generate a chain. From there, you'll install the packages you need for your application using Flora, our package manager. Let's say you want to add a new ERC20 asset. Run '`flora install erc20`' inside of the project directory, and a new token will be generated. To deploy the contracts to the chain, run '`saffron start`', and then '`saffron deploy all`'.

From there, you can access the administrative dashboard if you want to graphically interact with your blockchain. You can install packages from here, add accounts, see transactions, etc. much like Etherscan or another block explorer software. You will also have Remix and other smart contract development tools here as well.

From this backend, you can upload new contracts that you develop to Flora. You can also connect your chain to Clove to interact with other blockchain applications. To do this, you will host an IPFS node and connect to the distributed computing cluster as a server. You can then interact with other independant blockchains as a user, or to intiate chain-to-chain swaps. You can initiate swaps with the native Tau token that comes out-of-the-box with all blockchains.

Now, you can integrate your other software on top of the blockchain such as a public facing user interface, automatic transfers between chains, smart contract interaction, and even interacting with custom contracts that you deploy on other chains on Clove. All is possible with the Lamden Project.

Maintaining Distributed and Trustless System Architecture

While private chains on Lamden are built off of Ethereum which is a distributed and trustless protocol, the rest of the Lamden systems could be developed using standard centralized server-client architecture. However, while

this is potentially an easier method of execution, standard HTTP protocol is vulnerable to DDOS attacks, server failure, and centralized ownership of data that sacrifices the rights of the collaborators of the project[21].

Thus, the project will be deployed upon the IPFS protocol to establish a true peer to peer system that is distributed and trustless.

Lamden Tau, an agnostic digital asset

While payment channels across private chains will help facilitate app-to-app communication, there should also be away for users who are involved in the Lamden Project apart from a chain host or developer to easily participate and use the blockchain applications on Flora. Thus, we propose an agnostic utility asset known as 'Lamden Tau'. Tau are a digital asset that serve the universal asset of the platform itself, available for trade by users of the system who want to use a currency that use all private chains. Communication with the Tau chain comes out of the box for Lamden users, so they do not have to form independent relationships with other private chains to start transferring assets to one another. In this fashion, developers can start to integrate a digital currency market and trade into their blockchain applications immediately. By offering a native digital currency out of the box to all Lamden users, Tau increase the adoption rate of private chains, as more individuals are able to participate with the application coming from the Lamden community.

Token Sale (Token Scattering Event)

The Lamden Project will be funded by a Token Distribution Event (Token Scattering) of the Lamden Tau. These Lamden Tau will be available for purchase on the Ethereum main chain in the form of an ERC20 token. Said Lamden Tau will then be available for swapping onto the Lamden chain during a ceremonious process which signifies the milestone achievement of chain-to-chain communication on Clove.

Further detailing on the Token Scattering Event is included in organizational documents.

5. Conclusion

The goal of the Lamden Project is to provide a suite of tools that makes the rapid development and deployment of blockchains easy for the general population of developers. By modelling our tools based on the most popular tools used by developers today, we can capture a large and enthusiastic base of developers who want to get involved in blockchain but are unable to get over the initial hurdles.

Furthermore, by providing a centralized community, innovation can prosper and build upon itself, leading to newer and greater technologies, accelerating the industry as a whole.

Lastly, through Clove, we are able to connect all of these projects together on a single routing system that facilitates swap transactions so that private chains can retain the benefits of self-management and still take advantage of a larger ecosystem of great applications.

References

[1] Cryptofees, "Cryptofees: Bitcoin." Retrieved from: http://cryptofees.net/, 2017 (accessed 21 Aug 2017).

[2] PayPal, "PayPal: Transaction fees." Retrieved from: https://www.paypalobjects.com/digitalassets/c/website/marketing/na/us/ebooks/pdf/fees-page.pdf/, 2017 (accessed 21 Aug 2017).

[3] MasterCard, "MasterCard: 2017–2018 U.S. Region Interchange Programs and Rates." Retrieved from: https://www.mastercard.us/content/dam/mccom/en-us/documents/merchant-interchange-rates.pdf/, 2017 (accessed 21 Aug 2017).

[4] Visa, "Intra Visa Europe EEA: Multi-lateral Interchange Fees." Retrieved from: https://www.visaeurope.com/media/images/Intra%20VE%20EEA%20%40%202017_01_26-73-17763.pdf/, 2017 (accessed 21 Aug 2017).

[5] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2013. Retrieved from: https://github.com/ethereum/wiki/wiki/White-Paper.

[6] Stripe, "Stripe: Payments." Retrieved from: https://stripe.com/us/pricing/, 2017 (accessed 22 Aug 2017).

[7] Etherscan, "Ethereum ChainData Size (Geth w/FullArchive Sync)." Retrieved from: https://etherscan.io/chart/chaindatasize, 2017 (accessed 17 Aug 2017).

[8] BitInfoCharts, "Bitcoin (BTC) stats and information." Retrieved from: https://bitinfocharts.com/bitcoin/, 2017 (accessed 17 Aug 2017).

[9] MyBroadBand, "Bitcoin and Ethereum vs Visa and PayPal – Transactions per second." Retrieved from: https://mybroadband.co.za/news/banking/206742-bitcoin-and-ethereum-vs-visa-and-paypal-transactions-per-second.html, 22 Apr 2017 (accessed 17 Aug 2017).

[10] A. Sucharitakul, "squbs: A New, reactive way for PayPal to build applications," 2016. Retrieved from: https://www.paypal-engineering.com/2016/05/11/squbs-a-new-reactive-way-for-paypal-to-build-applications.

[11] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On bitcoin and red balloons," *CoRR*, vol. abs/1111.2626, 2011.

[12] pip 9.0.1, "pip: Python Package Index." Retrieved from: https://pypi.python.org/pypi/pip/, 06 Nov 2016 (accessed 21 Aug 2017).

[13] Yeoman, "Yeoman: The web's scaffolding tool for modern webapps." Retrieved from: http://yeoman.io/, 2017 (accessed 21 Aug 2017).

[14] npm, "npm: Node Package Manager." Retrieved from: http://npmjs.com/, 2017 (accessed 21 Aug 2017).

[15] C. Mason and R. Brown, "Entrepreneurial ecosystems and growth oriented entrepreneurship," tech. rep., OECD, 2014.

[16] T. McConaghy *et al.*, "BigchainDB: A scalable blockchain database," pp. 31–41, 2016. Retrieved from: https://bravenewcoin.com/assets/Whitepapers/bigchaindb-whitepaperDRAFT.pdf/.

[17] J. Poon, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016. Retrieved from: https://lightning.network/lightning-network-paper.pdf/.

[18] D. Ongaro and J. Ousterhout, "In search of an understandable Consensus algorithm," tech. rep., Stanford University, 2013. Retrieved from: https://raft.github.io/raft.pdf/.

[19] D. Mazières, "The Stellar Consensus Protocol: A federated model for Internet-level Consensus," tech. rep., Stellar Development Foundation, 2015. Retrieved from: https://www.stellar.org/papers/stellar-consensus-protocol.pdf/.

[20] M. Mahmoody, T. Moran, and S. Vadhan, "Time-Lock puzzles in the Random Oracle Model," in *Annual Cryptology Conference* (P. Rogaway, ed.), vol. 6841, pp. 39–50, Springer, 2011.

[21] Symantec, "Internet Security Threat Report," April 2016. Retrieved from: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf/.