Pallet

Protocol for Abstract-Level Ledger Ecosystem

Beyond Yet Another Chain:
IP Protocol for Future Internet of Values

Contents

Abstract

Today's blockchain encounters the following challenges: scalability and interoperability. To address these challenges, we propose Pallet, Protocol for Abstract-Level Ledger Ecosystem.

Pallet makes inter-chain transactions possible by decoupling the state of contracts from the blockchains and using Juries to execute contracts. Instead of forcing all contracts running on one chain, our paradigm is to have one Jury for one contract at all times. The decoupling of DApps from a particular chain boosts scalability. Furthermore, we propose a stake mechanism, which enables Pallet consensus protocol to minimize the perpetration through proper reward and penalty mechanism in the ecosystem.

Introduction

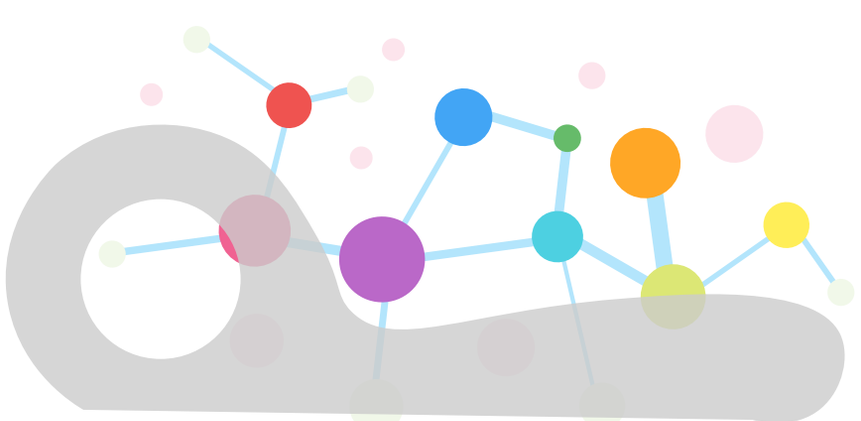
It is high time for today's money system to be revamped. The system today is slow, expensive, exclusive for someone, centralized, and in silos. As a result, the circulation of digital assets globally is not frictionless. Our vision is to enable an ecosystem where values can be transferred in an almost frictionless manner as information delivered on the internet today.

Currently, blockchain has gone beyond the geographical constraints and demonstrated the potential of the Internet of Value (IoV), shedding light on the future of the Internet of Value. However, blockchain today still faces these two challenges. First, the scalability of a blockchain remains elusive today. Second, the interoperability among different blockchains is lacking.

The creation of the genesis block on the Bitcoin blockchain in 2009 marks the beginning of an era of a decentralized and global digital currency. Ethereum extends the utility of blockchain with smart contract. The development of various distributed ledger technologies (DLT) is a precursor to the next-generation, the Internet of Value technologies.

Internet of Value

The Internet of Value is defined as enabling the future of "the exchange of any asset that is of value to someone, including stocks, votes, frequent flyer points, securities, intellectual property, music, scientific discoveries, and more" [Ripple 2017]. However, the current DLT faces two challenges in building the next Internet-of-Value world: scalability and interoperability.



Challenges

Scalability
Looking back on the history of DLT, Bitcoin is a distributed trustless network in which the tokens (values) flow. The consensus is reached by every node throughout the network to guarantee the correctness. The smart contract system proposed by Ethereum follows the same principle. Ethereum treats the network as a gigantic computer, where all the nodes execute the same procedure in order to reach the consensus on the state. As a result, Bitcoin traffic jam has become a popular search keyword since 2016. In December 2017, Crypto Kitties severely slowed down the Ethereum transactions. These phenomena reveal the problem of resorting to the consensus of the entire network. The congestion appears because all smart contracts share the same group of executing nodes, namely, the nodes in the blockchain network. As a result, a popular service may adversely affect the performance of other services.

Interoperability
Blockchains today such as Bitcoin or Ethereum are using full nodes as brute-forces trust-machines. These full nodes verify transactions on their respective chain without knowing anything outside their chains.

As a result, such a blockchain becomes a silo to itself, thus making the blockchain look more like an intranet today. Building a scalable and inter-operable internet-of-value system is challenging yet crucial nowadays.

Introduction to Pallet

In order to find an answer to the challenges above, we now introduce Pallet, protocol for abstract-level ledger ecosystem. Instead of introducing another blockchain adapting between every chain we want to co-work, Pallet works in a more efficient way to deal with scalability and interoperability issues at the same time.

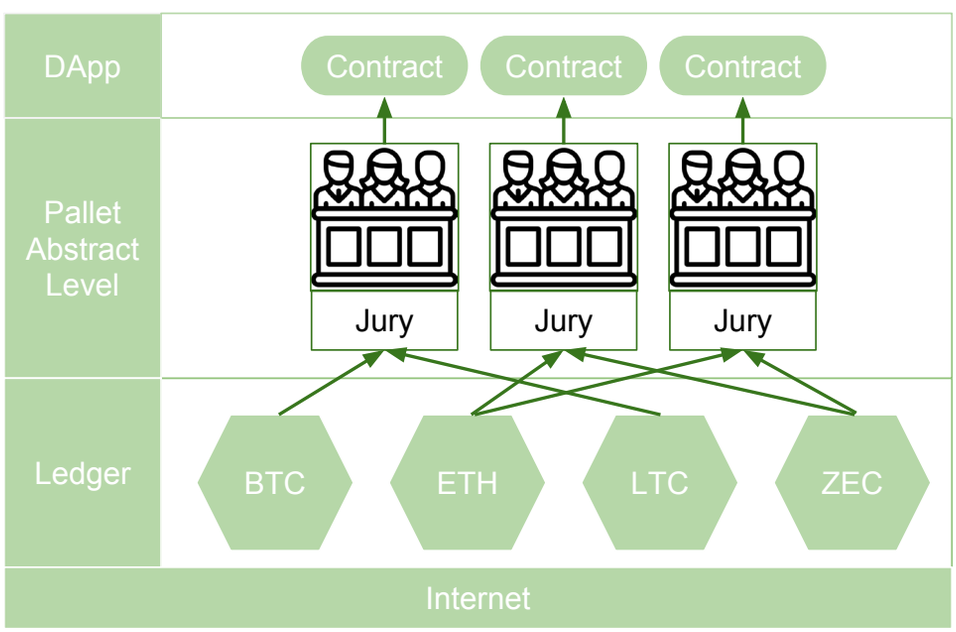


Figure 1 Concept of Pallet system. The contract's execution is on the abstract-level.

In figure 1, we can see that Pallet is an abstract-level smart contract protocol above blockchains, which divides value (or state of contracts) from blockchain. In Pallet's abstract-level smart contract, we only need a group of verifiers to execute a contract. This group of verifiers is called Jury, which consists of individual verifiers called Jurors. In our design, there is no need to reach consensus across the entire network anymore, which is more efficient and scalable.

However, benefiting from the division of value and chain via abstract-level, value transfer is no longer bound in a single blockchain. Value can be transferred to any other blockchain easily through our abstract-level protocol.

Here, we will use an example to show how inter-chain exchange in BTC and ETH is achieved through Pallet.

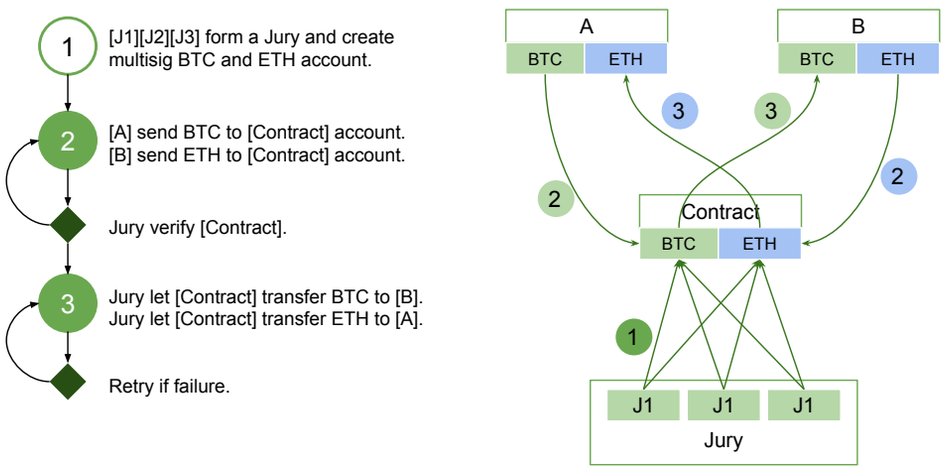


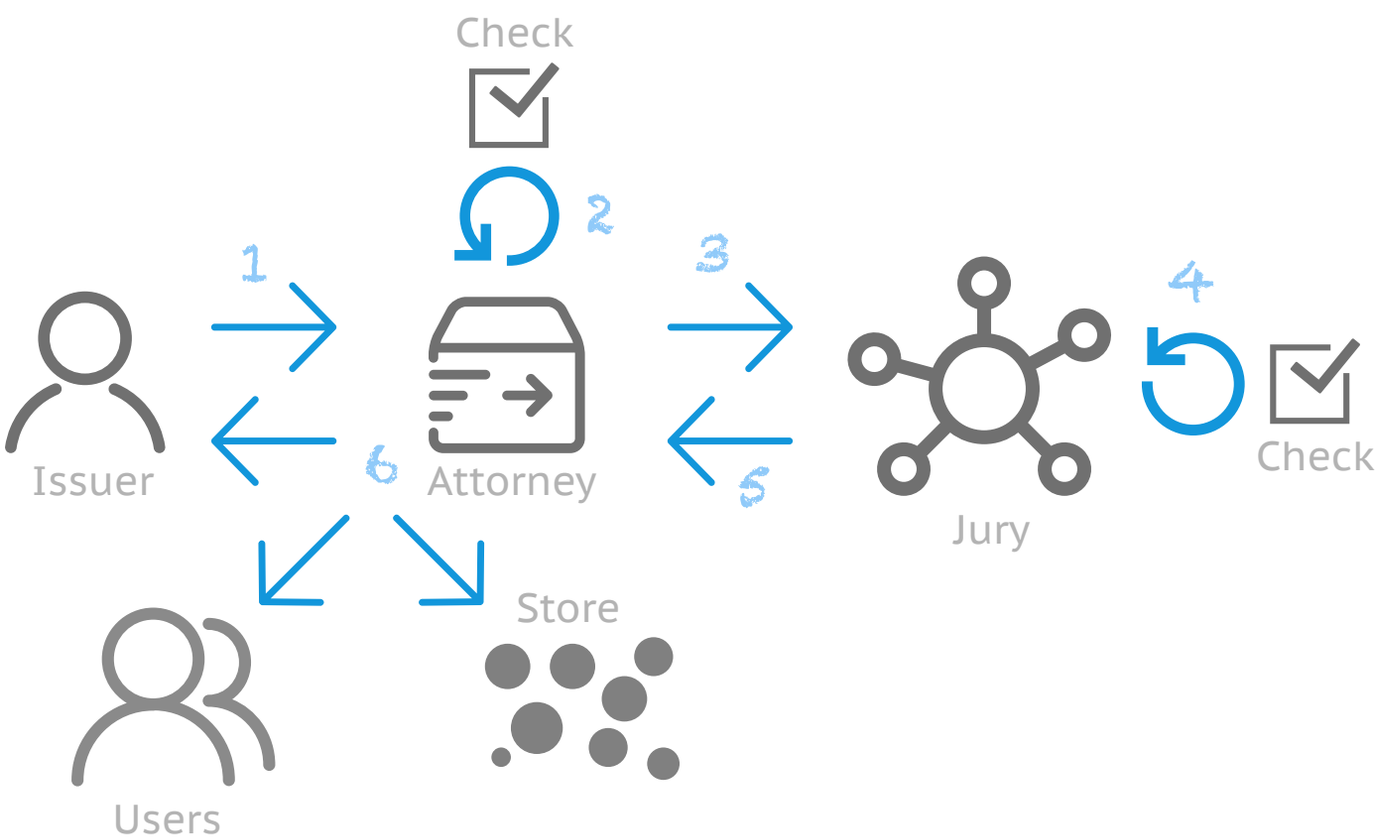
Figure 2: A example of inter-chain transfer between BTC and ETH through Pallet.

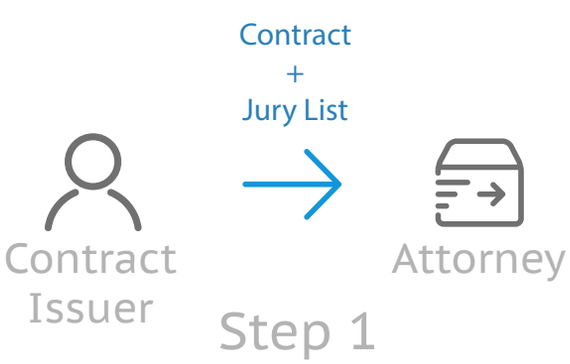
Figure 2 shows how we deal with an inter-chain exchange through Pallet. First, an exchange service provider selects Juror J1, J2, and J3 as Jury to execute the exchange contract. Then, Jury creates a multisig account on both Bitcoin and Ethereum network as the contract account. Second, if A and B want to exchange their BTC and ETH, they should send their token into the corresponding contract account. A sends BTC to contract account of BTC, and B sends ETH to contract account of ETH. Then, they may invoke the exchange contract, and the Jury verifies and tries to match the two order. If correct, Jury will update the state of contract. Last, A and B are allowed to withdraw ETH and BTC from the contract account according to the state of contract.

Pallet Protocol

In Pallet, there are two basic contract operations: deployment and invocation.

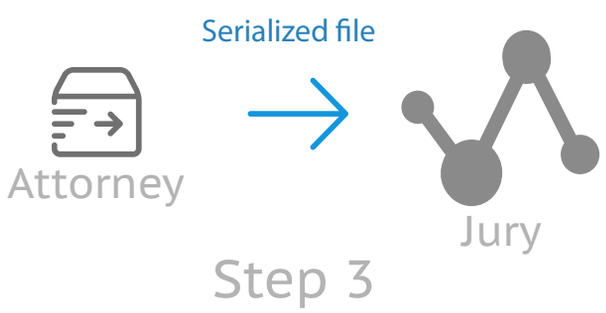
Contract Deployment





In Pallet, all kinds of services are created through contracts developed by contract developers. Once a contract issuer wants to deploy the contract, a group of Jurors will be selected as Jury. Then, the issuer needs to input the source code of the contract and the list of the selected Jurors into a client-side tool called Attorney.

Attorney compiles the received source code into an executable, then executes it to initialize the contract state and generate corresponding blockchain operations if needed.



The contract executable, the initial contract state, and the corresponding blockchain transactions are all packed into a serialized file. Next, the Attorney sends the serialized file to the selected Jury.

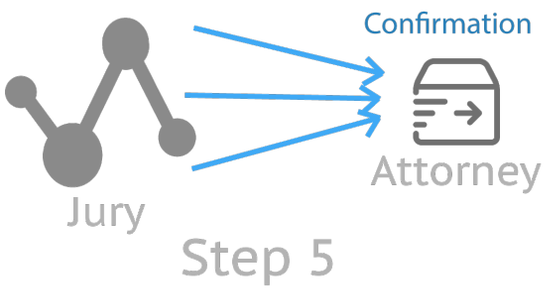
When the Jurors of Jury receive the serialized file, they act independently. The Jury parses the serialized file to get the contract executable, the expected initial contract state and the expected blockchain transactions for further verification.

The Jury then executes the contract executable without previous contract states as the Attorney does.

The contract execution also generates the initial contract state and corresponding blockchain transactions. The Jury then checks if they match the expected ones. If they match, the Jury signs the serialized file and the blockchain transactions respectively and sends the signed data back to the Attorney. Otherwise, the Jury rejects the contract deployment.

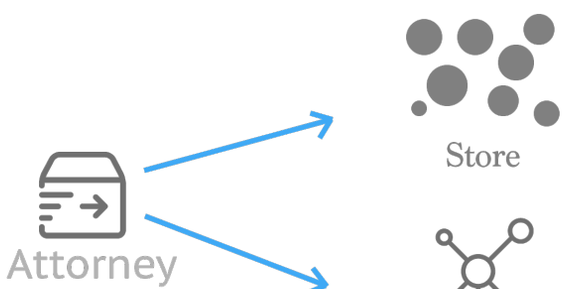
After the executions of Jury, the Jurors in the Jury broadcast their results to each other and reach a consensus of the initial contract state. Those Jurors keep the latest contract state internally.

Then, the Jury sends the signed signatures back to the Attorney.



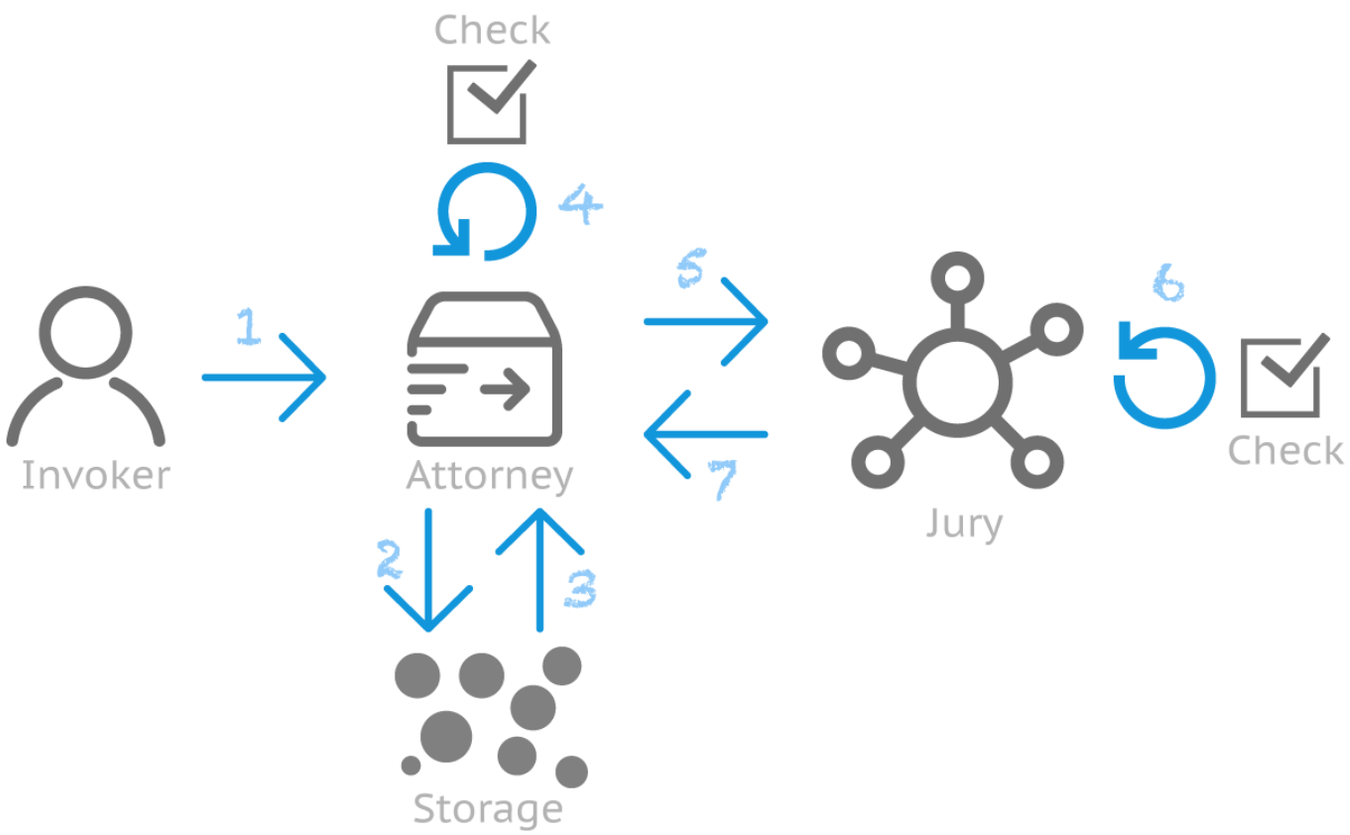
Step 5

Once the Attorney gets enough (2/3 of the selected Jurors for example) signatures for the serialized file and blockchain transactions, the Attorney broadcasts the blockchain transactions to the miners of the blockchain to make these transactions immutable. The issuer can also ask the Attorney to generate a new transaction that contains the signed serialized file and broadcast the transaction to blockchain to ensure the executing state is immutable. The Attorney also creates a unique ID for the deployed contract, and stores the contract executable and the list of Jury in a decentralized database by its ID.



Step 6

Contract Invocation



After the issuer deploys the contract, other participants can invoke the contract to interact with the contract. The following shows the workflow of contract invocation. To invoke the contract, the invoker inputs contract ID to Attorney. Then, Attorney can look up the database by the contract ID. The decentralized database will send back the contract executable and the list of Jury who were responsible for the execution of the contract.

Next, Attorney executes the contract according to the latest contract state and invocation parameters. This execution also generates the next contract state and the corresponding blockchain transactions. After packaging the contract ID, the contract states before and after the contract execution, the invocation parameters, and the blockchain transactions into a serialized file, Attorney then sends the file to the selected Jury.

When the Jury receives the serialized file, they also act independently. The Jury parses the file to get the packaged parts and immediately compares the contract state before the contract execution with the latest contract state it kept internally. The Jury will reject the contract invocation if the states do not match.

Otherwise, the Jury executes the contract according to the contract ID, the latest contract state, and the invocation parameters. The Jury then compares the generated next contract state and the generated blockchain transactions with the ones parsed from the serialized file. If they match, the Jury will sign the serialized file and transactions and keep the latest contract state internally. The signed data are then sent back to the Attorney. Besides, the Attorney will generate and broadcast a new transaction containing the serialized file to the beneath blockchain if necessary.

Transaction Fee

To provide incentive for Jury's execution, Jurors gain fees from contract participants by executing contracts. Contract participants need to deposit some Pal tokens as the transaction fees in the Pallet Distributor Contract (PDC). PDC is the contract that handles our default token Pal. The Jury only executes the contract after they verify that the Pal tokens have been transferred to PDC to ensure they will get paid. Since the Pal tokens are handled by PDC, paying Pal tokens as the transaction fees is fast (no need to wait for settlement on blockchains), inexpensive (no additional transaction fees on blockchains and we will also issue Pal token to incentive Jury), and convenient (native interaction between contracts in Pallet).

Consensus of Jury

A kind of the proof of stake (POS) mechanism is used here to reach the consensus of Jury. In the first place, there is a Jury 0 that executes an arbiter contract which can hold the stakes from the registered Jurors. When Jurors are grouped into a Jury, the consensus is based on the majority vote of Jurors where the vote of each Juror is weighted by its stake.

For example, when a Jury has five Jurors who deposit 1, 2, 3, 4, and 5 tokens as their stakes in the arbiter contract respectively, the Jury can reach consensus according to the result signed by the Jurors who deposit 1, 2, and 5 tokens because $1 + 2 + 5 = 8 > (1 + 2 + 3 + 4 + 5) / 2$. Or, the Jury can reach consensus according to the result signed by the Jurors who deposit 4 and 5 tokens because $4 + 5 = 9 > (1 + 2 + 3 + 4 + 5) / 2$.

So, as long as the attackers do not have more than half of the stakes in the Jury, the Jury can reach the consensus according to the contract correctly. However, once the attackers have more than half of the stakes in the Jury, these attackers may collude to reach a malicious consensus in this Jury. To keep the consensus secure, other Jurors or the participants of the contract can report the malicious results from attacker Jurors to the arbiter contract. The arbiter contract will re-execute the contract to verify if the results of the contract execution and/or the consensus are correct. If the results are incorrect, the arbiter contract will confiscate the stakes of the attackers and execute the return part written in the contract on how to return the funds if collision happens.

Therefore, the Jurors in the Jury 0 are very important because they are responsible for distinguishing the attacker Jurors and the good Jurors. We also apply the same POS concept to Jury 0. The Jurors in the Jury 0 are the major holders of the Pal tokens and their benefits are highly proportional to the price of the Pal tokens. Therefore, they have high incentives and profits to reach correct consensus when they execute the arbiter contract. Because if they reach a malicious consensus in the arbiter contract, the entire Pallet network is no longer trustworthy and the Pal tokens held by them would become valueless.

Recap

We can see that, based on the aforementioned Pallet architecture, the Attorney and the Jury can execute the contracts and interact with the underlying blockchains. Jury verifies the execution results sent from the Attorney and reaches consensus to perform the reliable contract execution. This also makes the execution efficient and scalable, since the consensus is generated by the Jury of this individual contract instead of all potential Jurors. To reduces cost of transaction fees and settlement latency, unlike most current implementation that has all contract executions on the blockchain, in Pallet, only the contract states are committed into the underlying blockchains with the regard of contract participants.

Pallet Attributes

Scalability

The key to reach scalability is multitasking. We use a participant-based design when achieving consensus.

In popular blockchain products, there are different consensus mechanisms. However, most of those algorithms essentially still based on the concept of consensus throughout the whole network. Every transaction consumes the computing power of the whole network for consensus. This approach is considered as an effective decentralized verification mechanism.

In Pallet protocol, instead of the entire network, the consensus is reached among one Jury, which is formed by a group of Jurors. Once it is reached, Jury will store the contract state internally. With this approach, we can effectively reduce the congestion of the whole network.

Interoperability

Contract in Pallet is not executed by the miners of blockchains, but by Attorney and Jury that are decoupled from blockchain. So, the execution model of Pallet can be applied to various different blockchains, including popular existing blockchains such as Bitcoin and Ethereum, as long as the blockchains allow users to put arbitrary data into the block if needed.

Take Bitcoin as an example. The arbitrary data that can be stored in the OP_RETURN field of a transaction with size up to 80 bytes. Having such a small size is a problem for the entire contract executable or contract state. To deal with it, we put the hashes of the packaged contract executable and states instead of the entire data into the blocks. This technique allows the Attorney to publish the contract progress to all blockchains in the same way.

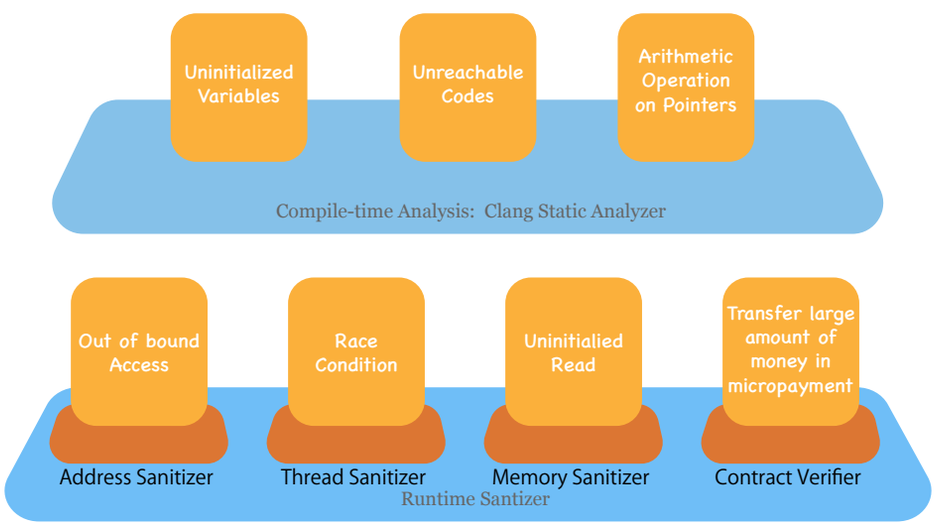
Pallet contract supports multiple blockchains. A contract developed for one blockchain (e.g., Bitcoin) can be easily reused on another blockchain (e.g. Litecoin), as long as there are well-defined system functions and libraries for Attorney and Jury to interact with the blockchains and generate the corresponding transactions.

This will significantly reduce the cost of contract development and make inter-communicate more easily. To benefit the developers, we will support some of the basic libraries and provide a detailed specification.

It is also possible to build a contract that can interact with different blockchains at the same time, so users can trade tokens from different blockchains in one Pallet contract invocation to make the inter-chain token exchange distributed, atomic, and immutable. Unlike EtherDelta that is slow and specific to the Ethereum blockchain, the Pallet contracts of inter-chain token exchange can be executed in a multi-tasking way by different groups of selected Jury, and the execution of the Pallet contract is fast because the contract progress confirmation in the blockchain is not necessary.

Other attributes

Security
The LLVM-based technology makes the contract execution more secure. Instead of trying to adopt new contract-oriented programming languages which limit how the contract developers to write the contracts, Pallet allows the developers to use their familiar programming languages, and utilize compile-time analysis tools, runtime sanitizers[1], and customized rules to check if there are any errors or threats in the contracts.



Security Guarantee

Security is also achieved through Pallet consensus protocol. Jury should deposit asset to arbiter contract as stakes for later execution. If Jurors of a Jury collude, the Jury 0 who are responsible for arbiter contract will re-examine the contract execution. If they are incorrect, the arbiter contract will confiscate their stakes as compensation.

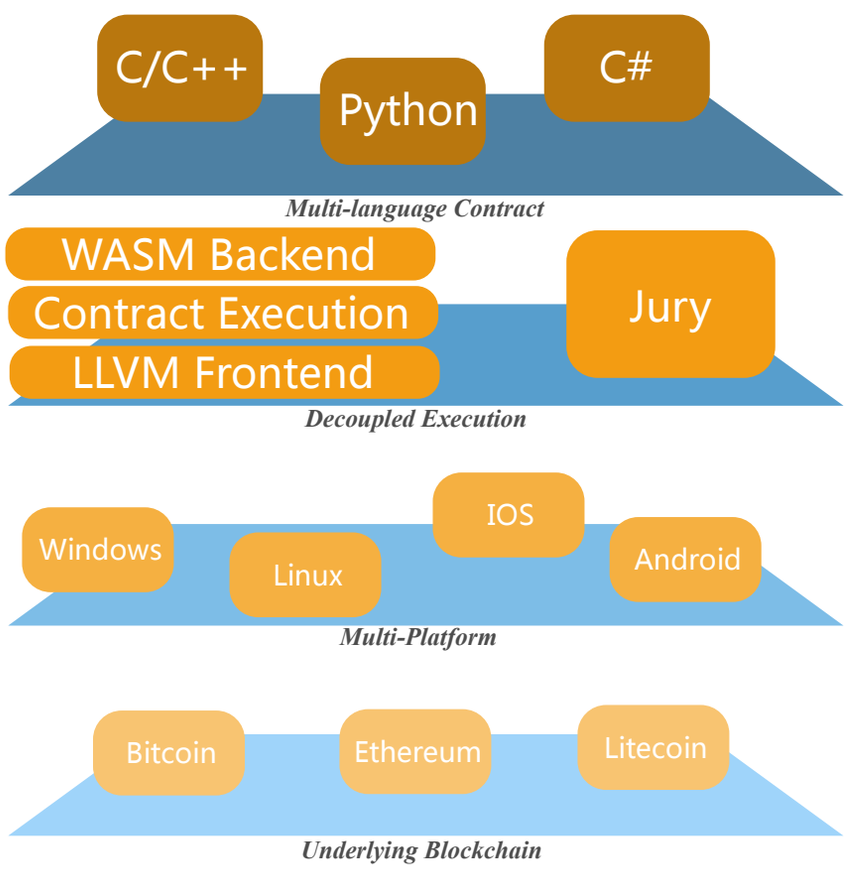
1 https://clang.llvm.org/docs/AddressSanitizer.html, https://clang.llvm.org/docs/ThreadSanitizer.html, and https://clang.llvm.org/docs/MemorySanitizer.html

Privacy
For the sensitive contracts, the contract issuers can encrypt the contracts, and only allow the contract participants' Attorneys and the selected Jurors to have the public key for decryption and execution. For the highly sensitive contracts, the contracts can be encrypted, and the computation on Jurors is on ciphertext, so that even Jurors cannot know the information in the contracts.

On-Chain/Off-Chain Interaction
Pallet allows one contract to act according to the real world events by retrieving data from a function we provide. This function will gather data and the information will be verified by Jury to achieve consensus.

Multi-platform and Multi-language
Because the execution model of Pallet contract is flexible, Attorney and Jury can be implemented in many different ways. In our first step, we plan to use LLVM and WASM as the core technology to build the contract executable and execute the contracts. Through the LLVM toolchain, Attorney can easily compile the contracts written in several programming languages such as C++ and Python with simplification and restriction into LLVM bitcode, and the LLVM bitcode can be further converted into WASM that can be executed in multiple platforms efficiently. So, Pallet contract is not only decoupled from the underlying blockchains, but also decoupled from contract languages and execution platforms.

As a result, developers can choose the language they are familiar with and the platform they need.

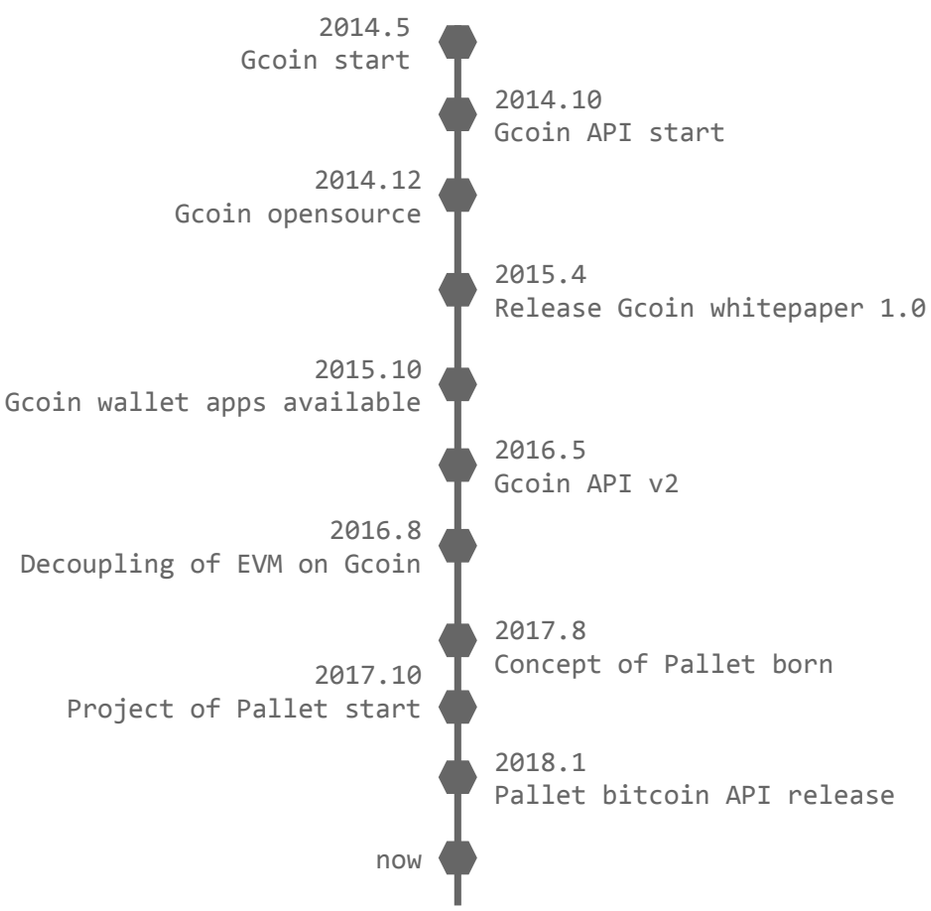


About Our Team

We are a group of passionate people who are fond of technology, and believe in the future vision of blockchain: the Internet of Value. We have developed Gcoin since Aug 2014. In the meantime, we have developed a series of tools[2] and apps for Gcoin and tried to promote blockchain technology for practical usage.

Staring in 2016, we tried to decouple Ethereum's smart contract system, make it adaptable to Gcoin, and even other existing chain like Bitcoin.

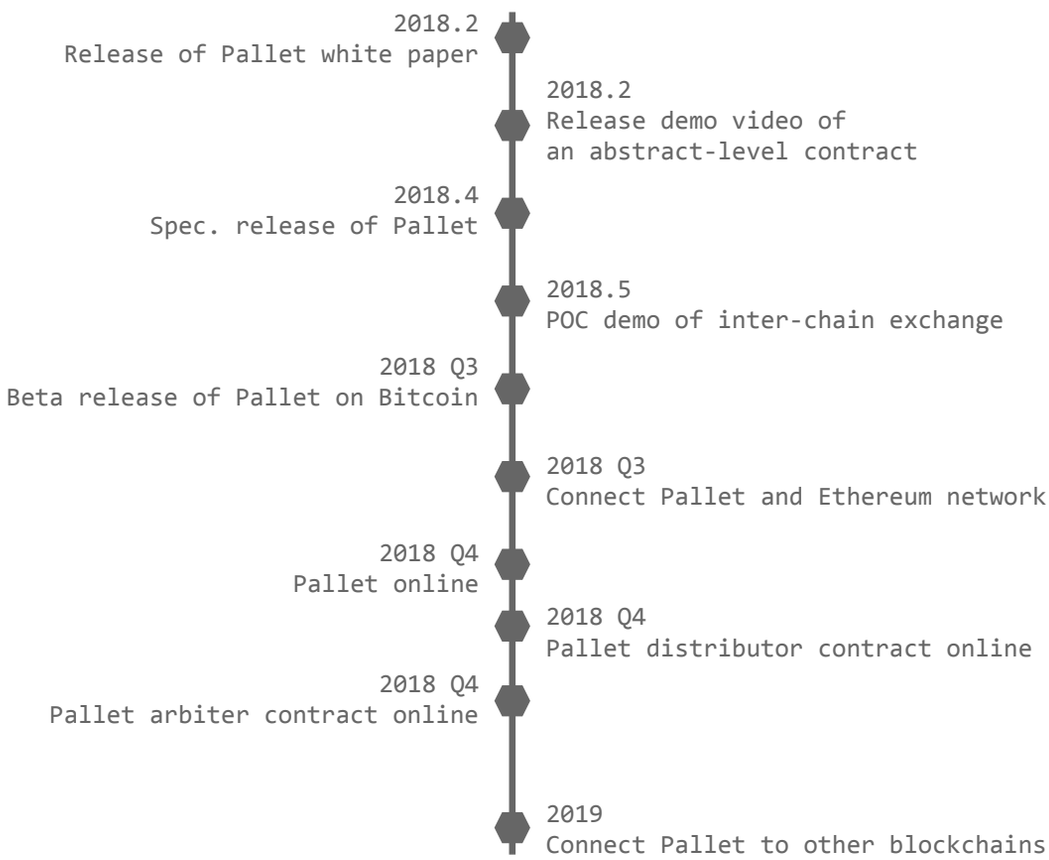
Meanwhile, we noticed the real potential of decoupling. It can do more than we think before, which is capable of overcoming the challenges nowadays and reaching the Internet of value. From then on, we start to research and design a protocol based on this concept. Thanks to the effort so far, we now introduce Pallet to start a new generation of the Internet of Value.



History timeline of our team

2: https://github.com/OpenNetworking/OSS-Federation https://github.com/OpenNetworking/OSS/
3: https://github.com/OpenNetworking/cotract_server https://github.com/OpenNetworking/oracle_server https://github.com/OpenNetworking/go-ethereum

Road Map



Conclusion

Pallet is an abstract-level smart contract protocol which decouples execution from underlying blockchains. As a result, execution of contracts can be more scalable and able to interact with different blockchain. Benefiting from leveraging LLVM and WSAM, contract in Pallet can not only be programmed in multiple languages but also be reused by existing tools to provide a secure and high performance execution.

Pallet allows users to trade on-chain and off-chain properties. To drive Pallet protocol, users can buy Pal tokens and use them as transaction fees to Jury, or users can become Jurors to earn Pal tokens.

Appendix

Token issuance demo code

Code 1: Pseudo code of token issuance

```
/* This is a pseudo code of a contract running on Pallet.
This contract will demonstrate how to issue a token in a
contract.
Some methods are defined in this contract, mint(), transfer()
and get_balance().
There are some predefined variables and methods provided by
Pallet contract APIs. */

init(args):
    // init(args) will be called only once when deploying.
    state = new_contract_state()
    state.set_issuer(current_user)
    state.set_empty_user_balance()
    set_contract_state(state)

run(args):
    // All invocations will start here.
    current_user = get_current_user()
    state = get_contract_state()
    param = get_parameters()

    if (args == “Mint N”) {
        return mint(N)
    } else if (args == “transfer N tokens to user U”){
        return transfer(N, U)
    } else if (args == “get_balance of user U”) {
        return get_balance(U)
    } else {
        return invalid_invocation(“Wrong arguments”)
    }
```

```
mint(n):
    issuer = state.get_isser()
    user_balance = state.get_user_balance()
    if (current_user == issuer) {
        user_balance[issuer] += n
        state.set_user_balance(user_balance)
        set_contract_state(state)
        return OK
    } else {
        return invalid_invocation("Permission denied.")
    }

transfer(n, receiver):
    user_balance = state.get_user_balance()
    if (user_balance[current_user] >= N) {
        user_balance[current_user] -= N
        user_balance[receiver] += N
        state.set_user_balance(user_balance)
        set_contract_state(state)
        return OK
    } else {
        return invalid_invocation("Insufficient token.")
    }

get_balacne(user):
     // Assume all balance infos are public.
     user_balance = state.get_user_balance()
     return user_balacne[user]
```

Glossary

The Internet of Value: The Internet of value will enable the exchange of any asset that is of value to someone.

DLT: Distributed Ledger Technology

Abstract level: Pallet is a light-weight protocol running on a higher level on the blockchain, where we call abstract level.

Attorney: A user-side client which is responsible for contract preprocessing and supports contract deployment or contract invocation command.

Jury: A group of chosen workers who are responsible for executing and verifying contract running on Pallet.

Juror: Contract verifier who is responsible for contract execution in the Jury group.

Pal Token: Pal token is used as the transaction fees of contract executions for Jury and is the native token of Pallet.

PDC: Pal Distributor Contract, a smart contract on Pallet which maintains Pal token.

Gas: The money pay for Jurors as an incentive to run contracts.