NULS 2.0
Philosophy of Design
V0.1

November 22nd, 2018
Content

1. SYSTEM Design Evolution: NULS 2.0 Microserver Platform - How is it Structured........ 3
   A. Design Reasoning and Explanation ................................................................. 3
   B. Plugin................................................................................................................. 4
   C. Microserver ..................................................................................................... 5
   D. Service Library............................................................................................... 7
   E. NULS Command Center ............................................................................... 8
   F. Libraries........................................................................................................... 10
   G. Interchange ..................................................................................................... 11
   H. NPS .................................................................................................................. 11
   I. Marketcap ....................................................................................................... 13
   J. Conclusion ....................................................................................................... 15

2. NULS 2.0 Cross-Chain Solution .............................................................................. 16
   A. Description ...................................................................................................... 16
   B. Key points in cross-chain design................................................................... 17
   C. An example for process of cross-chain transaction....................................... 18

3. Powerful Application: Chain Factory ................................................................... 20
   A. Why do we need an application like Chain Factory?.................................... 20
   B. What are the functions of the Chain Factory?............................................... 20
   C. How is the Chain Factory implemented?....................................................... 22
   D. Chain Factory and Basic Procedure ............................................................. 22
1. SYSTEM Design Evolution: NULS 2.0 Microserver Platform - How is it Structured

A. Design Reasoning and Explanation

Under the current software development habits, if you want to design and develop a large project, you will split the different services by splitting the modules. Different modules are handed over to different developers or development teams to design and implement. But you must define a common method of inter-module interaction, which is a simplified definition of the standard interface, as shown in the simplified Figure 1.

![SYSTEM Design Evolution #1: Modular Design](image)

This design worked of course, but since all modules were part of the same code structure, we still needed to process and compile all of the modules all at once. This required a lot of coordination effort to assure that all individual components were in a good state to perform preprocessing checks and compilation. It was no trivial task, because the system consisted of many modules plus dozens of components.
B. Plugin

Another problem arose, because although some customers didn't need a specific module, it was loaded nevertheless and consumed resources needlessly.

The solution proposed was to re-factor the whole code base in such a way that modules could be decoupled from the monolithic source-code, allowing them to be compiled and deployed independently, and also providing the possibility for the application to load them runtime-only if needed. The common term for this solution is called a plugin.

This architecture is not perfect. There are three main issues, the first of which is that testing and deployment still must be applied to the whole project. A second problem is that the user interface remains as an integral part of the code base, and lastly, only one development language can be used to enhance the system and make modifications, which is less than ideal.
The solution for this is to completely decouple the modules from the core project so that they become applications themselves. In this way, each module can be developed in any language, as long as they follow some prescribed conventions and protocols. Each one of these applications is called a Microserver; from now on Microserver and module will be interchangeable terms.

For this to work, at least one Microserver should be responsible for coordinating how other Microservers will behave and exchange information. This module is called a Manager.

This idea can work for most system types so why not make the Manager module even more abstract, and develop it to work for every use case? The resulting Manager module can be re-used to code other systems without modification.
Since every module is an application itself, could it be possible to run them on different servers? This may be useful if some modules require more computing power than others, or if a module requires special security features. How can we make this work?

The answer is adding another common Microserver to the system: the Controller, whose only functionality is starting and stopping Microservers according to certain predefined rules. If some Microservers run in other servers, then an instance of the Controller should be deployed on each server and configured to report information to the main server.
D. Service Library

So far this looks pretty good. But the number of modules is growing fast, and each one should know how to behave itself inside the system. The modules must understand how messages are sent and received, and in order to establish a common format, they must know how to connect to other modules and when. So, does all this work need to be duplicated inside each module?

The best way to attack this problem is to develop a piece of software that all modules need to incorporate into their code, in order to inherit all the common functions described, so that the rules don’t need to be written again. This piece of software is called the Service Base Library.

It should be noted here that in a multi-development language environment, the language corresponding to each module needs to develop a service library, such as Java service base library, C++ basic service library, Go basic service library etc.

Now all modules can behave themselves without duplicating any work!

Avoiding code duplication for developers is always a good idea, so let us stretch this idea even further...

Many systems share a lot of common functions:

• Storing data, whether in a file or in a database
• Some type of authentication
• Generating reports
• Offer translations to different languages

• Update the software

It seems logical that in order to avoid duplication of work for developers, we need to develop a standard set of modules to offer these common functionalities for different types of systems.

E. NULS Command Center

What about the user interface? Or speaking more broadly, what about external applications that are required to connect to the system? Do they need to connect to ALL Microservers at the same time?

For this we need to incorporate another module: the Connector. This module will collect the list of services and functions that the other modules provide and will expose them as though they were only one module to the outside world. This is called the API (Application Programming Interface).

The Connector Microserver is the only point of entrance, and therefore the only module the graphic user interface (GUI) application and other programs will need to
interact with, in order to take advantage of the whole system. The GUI will serve not only as a wallet, but also have several other functions, therefore a name change is in order: NULS Command Center.
F. Libraries

So now we have a GUI! But surely this isn’t enough to satisfy the necessities of blockchain innovators and supporters who wish to do more with the technology. Other official applications must be crafted, and NULS 1.x applications like the explorer, web wallet, etc. will need to be tweaked to work with the new platform. In order to reduce work for application developers (which is a common theme in this article) it would be reasonable to build libraries that allow external applications to connect seamlessly to the platform.

These libraries are pieces of code that provide the intricate details on how to manage network resources, ie. crafting messages, protocol handling, etc. so that developers will only need to focus on the problem that their application wants to solve, and NOT on how to integrate it into the platform. Over time these libraries can be ported to multiple languages to attract more external developers to the project.

The libraries are:

• Native Base Application Library
• Mobile Base Application Library
• Web Base Application Library
G. Interchange

External applications need to interchange messages with the NULS 2.0 system. How will this interchange work?

There are several ways to do this, and by far the most common pattern used (implemented in all other blockchain projects) is one called Request/Reply where the external application sends a request to the system, and the system sends a reply. This sounds like a fairly simple and logical solution.

Let’s dig a little deeper and assume that our application needs to show the balance of our address, in which case we would need to send a ‘get balance’ request, wait for a reply and then display the number returned. Since we surely want this to be updated constantly, we would need to send the ‘getbalance’ request every 5 seconds for example, to keep the balance displayed correctly.

In reality the operating system needs to add a lot of information to the ‘getbalance’ request so that the message can travel the internet, hopping from router to router (a process which appends even more information to the message) so that it can reach its final destination without getting lost along the way. We would need to repeat this process every 5 seconds just to keep the balance updated in our application.

H. NPS

Is there a way we can reduce the amount of traffic generated? Gladly, the answer is YES!

Rather than calling a function hundreds of times, we will simply subscribe to it instead. In this way, we can tell the system that we want to receive the balance only when it has changed.

This type of message exchanging is a heavily optimized variation of a pattern called Publish/Subscription and it is better suited to handle message interchange in modern
Let’s call this pattern: “NULS 2.0 Publish Subscription (NPS)” pattern.

At this point, we are able to build several types of systems by re-using the basic components presented and adding only specific modules, thereby shortening the time and cost for developers and entrepreneurs to create businesses around NULS.
I. Marketcap

Wouldn’t it be nice if we could just re-use the information provided from these systems to build new ones? For example, assume we want to create a system like Coin Market Cap to provide information about all tokens related to NULS. For a basic version we would need the price and the circulating supply. The price could be supplied by an exchange and the circulating supply could be supplied by the NULS 2.0 interface.

Instead of writing specific code joining these two pieces of information, we could just give developers one more tool: The Grouper. This module is in charge of gathering information from variances of NULS 2.0 and presenting it as though it was generated from our own system. In addition, our system could offer a function as part of its API. For instance, a MarketCap can be created, which can also be used by other systems.
The complete platform is depicted as shown:

The NULS’ ecosystem zoomed-out would look like something like this:

A beautiful mesh of fully interconnected systems.

It should be noted that the model presented here is the final vision and not all modules are necessary for the first production version, but many modules will be added over time.
J. Conclusion

NULS will no longer be just blockchain node software but a complete platform to build all kind of systems around (NULS is everything!). It will become a platform wherein systems can communicate with each other seamlessly, and this will allow businesses built around NULS to flourish a lot faster, and at the same time reduce development costs significantly. The sum of this optimization provides a very big competitive advantage among our peers.
2. **NULS 2.0 Cross-Chain Solution**

Blockchain has been introduced by media as the next Internet, which is quite reasonable. However, it is still far from being realized, and has a long way to go. For blockchain to become the next Internet, there is still a barrier to be crossed, which is, the value interchange. Currently, there seems to be two solutions:

1. One single blockchain completes the value interchange of all applications and users.
2. A general protocol can be used to achieve value interchange among blockchains.

The first option seems simple, but it is too limited to implement. NULS believes that the second option is a better one in the current situation and is working toward that direction. Cross-chain protocols and related facilities are the first step in this direction.

![Cross-chain Solution Diagram]

**A. Description**

Upgrade the NULS main-net and add functions responsible for docking with all blockchains; this achieves cross-chain communication in an open way.

Implementation based on the NULS module repository, means that blockchains (within the ecosystem) can add the cross-chain module, so as to communicate with NULS at the base layer.

For public blockchains such as Ethereum and Bitcoin which are not affected by NULS, it is necessary to implement the protocol conversion through a special mechanism,
and adapt the public blockchain protocol to the NULS cross-chain protocol to achieve the purpose of communication under a unified protocol.

Blockchains directly communicate only with the NULS main-net. The verification of the transaction is performed by the NULS main-net, and each parallel blockchain will trust the verification result of the NULS main-net.

Assets on a certain blockchain can be transferred to any chain that accepts the cross-chain protocols in the NULS ecosystem. It is also possible to achieve business interoperability between any two blockchains at a small cost based on mutually acceptable protocols.

B. Key points in cross-chain design

1. The NULS main-net adopts the Proof-of-Credit consensus mechanism and the Byzantine Fault-Tolerant mechanism to realize confirmation and packaging of cross-chain transactions, so as to achieve decentralization, high performance & security.

2. Each node on the NULS main-net connects to multiple nodes in multiple blockchains. Because the protocol is the NULS cross-chain protocol, which is uniformly defined, it is possible for a single node to connect multiple nodes in different blockchains simultaneously.

3. The NULS main-net provides the chain management mechanism to manage all equal-level blockchains registered on the NULS main-net. The contents of the registration include chain information, asset information, cross-chain deposit, etc.

4. When assets from other chains are received in a blockchain, corresponding assets need to be generated in this chain. Tokens on different blockchains are stored in other chains in the form of assets.

5. The details of the assets in a blockchain transferred from other chains will be stored in the NULS main-net. When the asset is transferred out of the blockchain, it will be verified, and illegal assets will not be allowed to be generated from the blockchain. Malicious blockchains are handled through community mechanisms such as suspending cross-chain, stopping cross-chain, forfeiting deposits, etc.

6. The NULS main-net will provide the API manual. Developers can develop their own wallet, explorer, web wallet and other tools according to the manual.

7. NULS main-net provides protocols for applications to extend, which can be used to develop DApps and optimize cross-chain protocols.
C. An example for process of cross-chain transaction

1. ChainA initiates the transaction TX_A, transferring 10 TokenA to AddressB on ChainB. (Create transaction)

2. By switching protocol, ChainA uses TX_A to generate a new transaction TX_A_NULS in compliance with NULS main-net protocol, packaging TX_A on ChainA and confirm N blocks. (Create transaction in compliance with NULS main-net protocol)

3. ChainA broadcasts TX_A_NULS to all connected nodes on NULS main-net. (Broadcast transaction)

4. Notified consensus nodes on NULS main-net acquire TX_A_NULS_Hash and TX_A_Hash from TX_A_NULS, sending them back to all connected ChainA nodes. If a certain NULS main-net node has more than 51% connected ChainA nodes confirm the Hash, it will confirm the transaction, create the signature and broadcast. (Confirm transaction)

5. After 80% of the nodes on NULS main-net confirm, the transaction TX_A_NULS will be packaged. (Package transaction)

6. NULS main-net broadcasts TX_A_NULS to all connected nodes on ChainB. (Broadcast transaction)

7. Notified consensus nodes on ChainB acquire TX_A_NULS_Hash from
8. ChainB nodes send TX_A_NULS_Hash back to all connected NULS main-net nodes. If a certain ChainB node has more than 51% connected NULS main-net nodes confirm the Hash, it will confirm the transaction and broadcast. (Confirm transaction 1)

9. If that node happens to be a producer of the last 50 blocks of ChainB, TX_A_NULS will be transferred to TX_B by protocol, and then signed and broadcast. (Confirm transaction 2)

10. If 80% of the producers of the last 50 blocks has signed TX_B, it can be packaged. After the package is done, AddressB will record 10 TokenA. (Package transaction)
3. Powerful Application: Chain Factory

A. Why do we need an application like Chain Factory?

The development of blockchain is very similar to that of the Internet. The blockchain itself is an upgraded version of the Internet, which can transfer value fairly.

There are many applications on the Internet, and there is no interference between them. However, in the blockchain world applications in the same chain will compete for resources, which is an urgent issue to be optimized.

In the blockchain world, there are only two ways to achieve business isolation, i.e. DAG and multi-chain. Both schemes should be tried and can even be integrated. One possibility is to integrate DAG in the multi-chain ecology. Moreover, there are a lot of uncertainties for DAG, while multi-chain is easy to understand. Every business has its own chain, maintains its own data, and does not intersect with irrelevant business, so there is no competition for resources.

Can we use blockchain like the Internet now? To start a terminal to access the blockchain network, you can choose to participate in any chain application. In this way, applications are isolated from each other's data and business, but the value is interoperable. We think this can be achieved. The Chain Factory is designed to achieve single-node and multi-chain parallelism, value interoperability and business isolation. Based on the above advantages, it also lowers the threshold of using blockchain for the builders (enterprises and individuals). With the help of the Chain Factory, you can even implement customized blockchain applications or business systems without concerning yourself with the underlying technology of blockchain.

B. What are the functions of the Chain Factory?

The Chain Factory itself is an application-oriented blockchain project, which runs in a decentralized way in an open network.

1. Module Repository

   The NULS team and community contributors will develop a series of blockchain basic functional modules based on the NULS 2.0 microservices architecture platform, to meet the requirements of different users for the underlying blockchain. They will also open the source code and operate the program. Anyone in the whole ecosystem can customize its own blockchain using the modules in the module repository.

2. One-Click Chain Creation
To create a blockchain in the main-net of the Chain Factory, you can fill in the personalized settings of the chain in the operation interface, and then just send a transaction by clicking the button. This chain will notify all nodes in the Chain Factory, and all of them can choose whether to run the chain, or start the chain at any time.

The chain launched by the NULS Chain Factory has complete blockchain properties and independence. You can generate independent running programs through the Chain Factory at any time, which are stripped from the Chain Factory main-net. The Chain Factory only provides technical implementation without limiting the user's operating environment.

3. **Free Choice of the Running Chain**

   After operating some chains in the Chain Factory, nodes can stop any part of them at any time.

   There will be different benefits from these running chains, and you can decide which chains to run, based on the value and business situation.

   Because the consensus mechanism may be different, you need to pay attention to the competition of hardware resources when multiple chains run on the same node.

4. **Value Interoperability**

   The chain launched by the Chain Factory is accessed to the NULS 2.0 cross-chain ecosystem by default. Assets in the chain can be circulated in the whole ecosystem through cross-chain protocols, and the application scenarios will be more abundant.

5. **Business Extension**

   The chain launched by the Chain Factory supports business extension. By customizing modules, you can extend your own blockchain. The nodes that run the chain can download the required extension modules from the module repository and start running at any time.
C. How is the Chain Factory implemented?

Chain Factory is an independent public blockchain built on the NULS2.0 platform. Each module can run multiple chains and achieve isolation for business and data.

Chain Factory will integrate Docker to automatically create containers, install images, and run services on the server.

Chain Factory will serve as a blockchain cloud platform that is available to anyone who wants to enter the blockchain world.

D. Chain Factory and Basic Procedure

1. What’s Chain Factory?
   
   Chain Factory is an application blockchain built based on NULS modules. This chain uses PoC (Proof-of-Credit) consensus mechanism and communicate with NULS main-net through cross-chain protocol.

   Each node of the Chain Factory is a cloud server.

2. Create a blockchain
   
   - Chain Factory provides the interface for users, such as the import of keystore. It guarantees that there are enough NULS tokens in the imported account for the cross-chain deposit.
   - **Select Modules**: Network, block management, transaction management, account, consensus, event bus, etc.
   - According to the modules selected, there can be different configurations (such as PoC consensus or PoW consensus). The system operating parameters (including incentive mechanism) must be properly set before the blockchain is created.
   - Set up the first block.
   - Fill in the introduction and creating method of the chain (1. embedded in the application chain, 2. containerized).
   - Pay the deposit, broadcast transaction.
   - Wait for the transaction to be confirmed.

3. Operate the blockchain
   
   - After the transaction of chain creation is confirmed, the information of the chain will be displayed in all wallets, and an operation portal will be provided to operate the blockchain.
• Operate the blockchain by one of the modes (1. Operate multiple chains directly in the module; 2. Generate mirror image, create container, load the mirror image and operate)
  o If the modules required by the chain exist in the local wallet, operate the chain directly.
  o If the modules are not in the local wallet, they will be automatically downloaded from the module repository. After the module signature is verified, the chain will be operated after the modules start.

• Features when blockchain is operating:
  o Mode 1: data isolation, network connection sharing, business independence, general UI
  o Mode 2: complete node isolation, general UI

• You can stop operation at any time at the provided operation portal, and uninstall the chain (stop operation, clear the data).