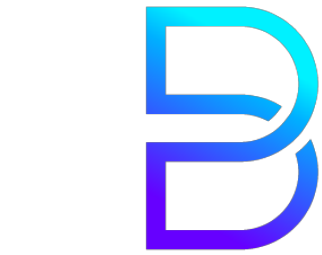

BIFROST

We Make DApps Work

Version 1.0

Disclaimer

The website and white paper prepared by Bifrost is for elaboration, description and for informational purposes only, and is not an offer or a solicitation to buy or sell any securities or to invest in any financial instruments. The registration on the Bifrost platform permits access to the services provided by the platform as detailed herein. Registration is not intended to afford the holder any rights in, or claims to, any of the assets of Bifrost or to in any way share in any profits that the platform may achieve. Interested parties acknowledge agreeing to the Consent to Use Electronic Records, Privacy Policy and Terms and Conditions. The white paper or content on this webiste is subject to change at any time without notification. The white paper and website describes the current plan and vision for the Bifrost platform. While we intend to attempt to realize this vision, please recognize that it is dependent on quite a number of factors and subject to quite a number of risks. We do not guarantee, represent or warrant any of the statements in the white paper or website, because they are based on our current beliefs, expectations and assumptions, about which there can be no assurance due to various anticipated and unanticipated events that may occur. Please know that we plan to work hard in seeking to achieve the vision laid out in the white paper and website, but that you cannot rely on any of it coming true. Blockchain, cryptocurrencies and other aspects of our technology and these markets are in their infancy and will be subject to many challenges, competition and a changing environment. We will try to update our community as things grow and change, but Bifrost v0.9 undertake no obligation to do so. Interested parties acknowledge that the Bifrost platform, as described herein, may never in fact operate as intended. Parties also acknowledge that all services and scope of work proposed in this white paper is subject to any licensing required. The information and graphical content contained in the white paper and website should not be construed as a guarantee and is subject to change at any time without prior notification. The information contained herein is intended for familiarization, and should not be utilized or reproduced in any form in full or part. The white paper has been prepared to the best of our knowledge and research, however it should not be relied upon for any future actions including but not limited to financial or investment related decisions. The company, founders, advisors or affiliates shall not be liable for any losses that arise in any way due to the use of this document or the contents contained herein. The content, both written and graphic may be historical or forward looking and therefore should not be relied upon. The content is based on assumptions and therefore uses words such as ‘expects’, ‘intends’, ‘will’, ‘can’, ‘should’ or similar expressions. The assumptions drawn in this document are based on past trends and data from third parties and other sources, which were believed to be reasonable at the time they were made. However, they still involve unknown risks and uncertainties, as it is impossible to predict anything outside of our immediate control including economic factors. Individuals and investors are requested to carefully consider the risks, costs and benefits of acquiring the token through this crowd sale as opposed to through a third-party exchange, once operational.

Contents

1 Introduction

1.1 Motivations and problems

We have high expectations with positive evaluations about future innovations that blockchain will bring. Bitcoin demonstrated the possibility of the functions of money through decentralization, and Ethereum used smart contract with technological advances to create a foundation on which many applications could actually operate. Smart contracts and decentralized applications (DApps) enable various services on the blockchain. As DApp services have increased and diversified, the blockchain ecosystem has matured. Given the superior incentivization structure, stability, and transparency under the decentralized network, DApps can be a game changer for the entire software business in the near future.

However, the speed of DApp's development is significantly below the public's expectations, and there are even doubts about the effectiveness of the blockchain technology itself nowadays. Particularly, limitations are found, for example, Ethereum network with more than 1,000 DApps is unable to provide effective technical support due to scaling issues. In addition, efforts to solve all the problems about decentralization, safety, and scalability (called Trilemma) at once in the protocol level, have led to the phenomenon of "Protocol Fever," in which the number of protocols is larger than that of DApps. Given the fact that DApp developer's first decision is to select a blockchain network protocol to provide DApp services, the current phenomenon of Protocol Fever is a major impediment to stable DApp development environment. The presence of various protocols and diverse DApps deepens mismatches with each other's needs.

Despite the structural problems of this blockchain technology, it has revolutionized beyond the national and industrial sectors, and most of them have been commercialized using their own managed blockchains (or private blockchains). In the field of copyright and logistics in particular, the private blockchain is pioneering new areas with unique data reliability and performance improvements. For instance, "IBM Food Trust," a blockchain-based food tracking network that was commercialized in October 2018, is a system that utilizes its own managed blockchain by sharing all details such as food origin and transportation records.

The ecosystem of dynamic DApp environment is essential for the development of blockchain as an innovative service. As with many other high-tech fields, the DApp field has difficulties that must be surmounted to construct its ecosystem and establish stable business service platforms. In this chapter, based on the technical environment of the blockchain described above, we define the problems that may arise in the DApp development environments, present the prerequisites for its solutions, and ultimately show Bifrost solutions for the DApps to work.

The following three problems are found in the DApp development environments:

The first problem is "the platform risk." DApp developers have to choose a blockchain mainnet protocol (e.g., Ethereum, EOS) to provide their services. Numerous mainnet protocols have been developed, and many others with different emphasized functions are currently ready to go public. All the mainnet protocols are unique and have different advantages and disadvantages; however,

none provides a perfect development environment for DApps. Nonetheless, developers have to choose one and accept the uncontrollable risk that comes from the forthcoming projects of the chosen protocols.

Second, DApps have too many different requirements for blockchain mainnets. Since DApps provide different services, the system requirements can be varied. However, no single mainnet protocol is able to accept all demands. For example, crypto games must handle a number of transactions in their domains and transfer token rewards safely. That is, crypto games need scalability, stability, and decentralization at the same time, which is almost impossible in a single blockchain network.

Third, developers can choose their own managed blockchain (e.g., private blockchains), but the networks have limitations. Many research studies are being carried out to obtain more flexibility and scalability using private blockchain systems as a part of mainnet blockchain networks. However, private blockchain systems have more difficulties in verifying safety and reliability to construct DApp ecosystems than externally operating blockchain networks.

1.2 Approaches for efficient solutions

We consider the following two approaches to get efficient and practical solutions to the three problems outlined above in the DApp development environment.

First, no single blockchain mainnet protocol can be the solution for a DApp platform. Since we do not have "the perfect blockchain," it is inefficient and risky to depend on one blockchain mainnet as a platform.

Second, adding OS or abstraction layers increases overheads only. Implementing improvements via mainnet blockchain extension may overload the entire system and limit scalability.

1.3 Bifrost method

The premises above suggest that we need a brand new platform, where we can use existing blockchain systems as building blocks to achieve the strength of each system and organize environments for independent DApp services at the same time. That is, if we can make a Dapp platform that combines multiple blockchains that provides the reliability and safety as well as some customizability with using internally operating blockchain network, the problems of the DApp development environment can be solved.

We introduce "Bifrost," the new standard DApp platform. Bifrost focuses on DApps' practical services on blockchains. Bifrost provides a multiple blockchain platform where we can compose existing qualified blockchain systems as components and restructure DApps so that they can run on the combined blockchain systems. Inside of the platform, Bifrost supports the entire process, from DApp deployment to DApp operation. In the deployment phase, Bifrost splits DApps' codes into pre-designed blockchain parts and restructures them automatically. After the components are deployed in each position, Bifrost serves as a "Linker," which communicates

with deployed components and supports safe execution in the operation stage. BIFROST offers an improved DApp platform, which has the safety and the high performance of multiple blockchain systems.

BIFROST has flexibility to prepare for the uncertain future of blockchain mainnet protocols. BIFROST has the following characteristics:

1. **Blockchains as building blocks.** BIFROST constructs a parallel environment by combining multiple blockchains, which can come together and form a federation.

2. **Container applications.** One blockchain can form an independent Container and each Container can have a different setup. That is, the creation and destruction of the Containers are dynamic, and performance tuning, such as increasing the speed of transactions or enhancing anonymity, can be applied to the Containers. In addition, a new approach to managing the Containers is possible. For instance, an independent Container can be operated and destroyed after execution if the service must not be recorded.

3. **Multi-component DApp reconstruction.** DApps, including smart contracts on the BIFROST platform are reconstructed to have appropriate components for each blockchain and run simultaneously. Therefore, DApp can implement the high performance as well as the safety verification of multiple blockchains. The reconstruction technology of multi-component DApps is working for the existing DApps, and it will be upgraded to support the full flexibility of BIFROST by introducing a new high-level programming language.

BIFROST has the real potential to expand the horizons of DApps. DApps can ultimately overcome platform risk from the blockchain mainnet protocol and utilize the strength of more than one blockchain by using combinations.

2 BIFROST System

2.1 Design

DApps' dilemma or trilemma. A number of blockchain systems are being introduced, each with their own unique features. DApp developers are trying to find a blockchain system that matches their own requirements perfectly. However, it is not an easy task. Once a DApp is built on the a certain blockchain system, it can reap the benefits of the blockchain's advantages (e.g., high scalability), but it must also tolerate its disadvantages (e.g., limited safety). Consider a DApp used to sell digital contents via a blockchain system. Some blockchain systems are suitable for operations for asset transfers or ownership publications. On the contrary, some blockchain systems are preferred for inside auctions or negotiations. Under this dilemma (or trilemma), a DApp service provider is faced with the challenge of picking a blockchain system that optimizes the performance and security of its service.

Don't reinvent the wheel. Most of the newly proposed blockchain systems have a common, ambitious approach: building a completely new blockchain system that solves all the problems at once, particularly those related to scalability. However, every DApp has its own goals and needs. A single blockchain system cannot satisfy all the requirements. Furthermore, no blockchain system is free from intrinsic and technical tradeoffs. Thus, we are still in a vicious circle, where a new solution causes another problem.

Our approach: cherry picking in a good way. We present a new DApp platform, Bifrost, as a practical solution to the problems of DApps. DApp service providers and developers are confronted with two challenges: (1) selecting a mainnet blockchain for their service and (2) meeting the DApp's (paradoxical) scalability and security requirements. We tackle the fundamental problem itself: DApps just need a platform that satisfies all their different requirements. Therefore, we focus on how DApps can use desirable features from the existing blockchain systems rather than on making another mainnet blockchain. Bifrost combines multiple blockchain systems into a unified smart contract environment. It also automatically restructures the DApp itself to run directly on top of the multiple blockchain system.

Bifrost is a DApp platform that supports a DApp from development to operation. A DApp service consists of a core smart contract code and interfaces to web, external storage, and users. A smart contract is represented as a set of variables (state) and functions. Thus, the features of a DApp can be divided by functions. In Bifrost, a DApp developer creates the DApp service in a conventional way but s/he may also additionally indicate the target blockchain system for each function in the code. In the deployment phase of the DApp, Bifrost maps the functions to their targets. It then splits the smart contract code and adjusts (and transpiles) each split component to its target blockchain system. However, to enable the seamless interoperation of the split DApp, a platform also needs to support the execution of the DApp as well as the sophisticated conversion of the smart contract. To do so, Bifrost automatically generates a description for interfacing the components. In the operation phase, it then bridges them to run as a single DApp.

Design goals. The design goals of Bifrost are as follows:

- Do not let a DApp be bound to a single blockchain system.
- Provide a unified smart contract environment on top of multiple blockchain systems.
- Let a DApp run directly on the combined environment and reap the benefits from the each blockchain systems.
- Minimize the overhead of the multiple blockchain DApp. (Provide the performance of native smart contracts for each blockchain system.)
- Present a new program language to extend the smart contract program model for the multiple blockchain environments.

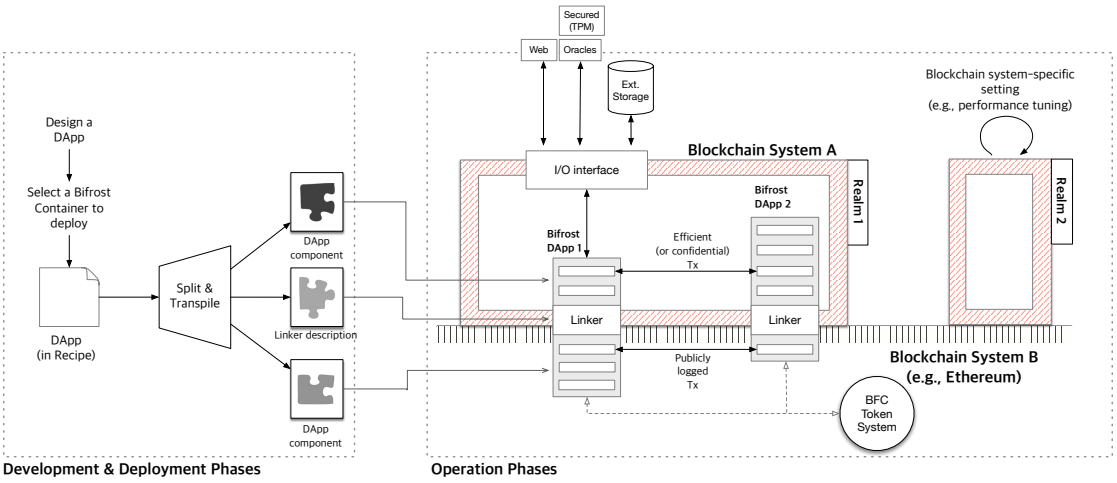

Figure 1: BIFROST process

2.2 Overview

BIFROST provides a DApp platform on the top of the multiple blockchains. In particular, to achieve the goals in §2.1, BIFROST works on the lifecycle of the DApp. It supports the DApp from the development phase to the operation phase. In the conventional integration of heterogeneous systems, one usually adds an extra abstraction layer, such as a virtual machine or chain, to establish a unified environment. However, the layered approach causes computation overhead and restricts the use of subsystem-specific features. BIFROST provides a more fundamental solution with advanced programming language techniques. A DApp in BIFROST is developed for the multiple blockchain system with little effort, and it is restructured to the native form for each blockchain system in the deployment phase. For this purpose, BIFROST splits a DApp into different smart contracts for each system in multiple blockchains, and then transpiles (translates to a different language) each one to the native smart contract form. Thus, the multiple blockchain system executes each DApp component directly without overhead. Since BIFROST combines multiple blockchain systems, the DApp can securely use the token systems of existing blockchain systems and execute functions efficiently with the component on manageable blockchain systems (e.g., private blockchain) at the same time. Moreover, this process can be performed by extending the existing smart contract code, such as Solidity in Ethereum. However, to support the new program model of the multiple blockchain system, we present a new DApp programming language, called "Recipe."

How BIFROST works. The multiple blockchain environment is composed of BIFROST Containers. A set of BIFROST Containers establishes a blockchain network and connects to the existing blockchain networks, such as Ethereum. For a DApp, the BIFROST process consists of a (1) development phase, (2) deployment phase, and (3) operation phase. In the development

phase, a DApp developer selects the multiple blockchain combinations that fits his/her requirements, and writes the smart contract code in Recipe, indicating which functions and variables are matched to which blockchains in the smart contract code. In the deployment phase, Bifrost splits and converts DApp codes into multiple components and generates the description for the interoperation of the components. Each component code is transpiled to the program language of the smart contract of the blockchain system. Bifrost then automatically checks the correctness of each component. The verified component can be deployed into each blockchain system. In the following operation phase, the Bifrost Container supports interoperation between the split components from a DApp.

The overall process. Figure 1 shows the overall process of Bifrost. The detailed process is as follows.

1. The developer designs a blockchain service and DApp and selects a Bifrost Container to get a desirable combination of multiple blockchains.
2. The developers writes the smart contract code for the DApp in Recipe.
3. The developer marks the target blockchain in the smart contract code.
4. Bifrost analyzes the smart contract code and splits it into components for each blockchain. It also generates the description, which contains the interface details to connect the components.
5. Bifrost deploys the components and the description to each part.

In the operation phase, Bifrost supports the interoperation of the DApp as follows:

- Each component can employ the unique features of the underlying blockchain.
- The Bifrost Linker interconnects the data and control flows between the components.

Bifrost does not recreate a new blockchain system, but puts the existing blockchain systems into the placeholders. The whole Bifrost system itself runs by Bifrost Nodes. Each Node is connected to the selected existing blockchains and has a Bifrost Container. For clarity, we first describe the elements of the Node system and then explain the DApp building process with Recipe, and the Linker operations.

2.3 Node architecture

Bifrost Node systems have an architecture to build an execution environment of the multiple blockchain systems to DApps. When connecting different blockchain systems, we need to handle the layered interconnection, since a single blockchain system already consists of multiple functional layers, such as consensus, network, and smart contract modules. Hence, the architectures of Bifrost Node systems are represented as stacks of supporting modules and blockchain systems.

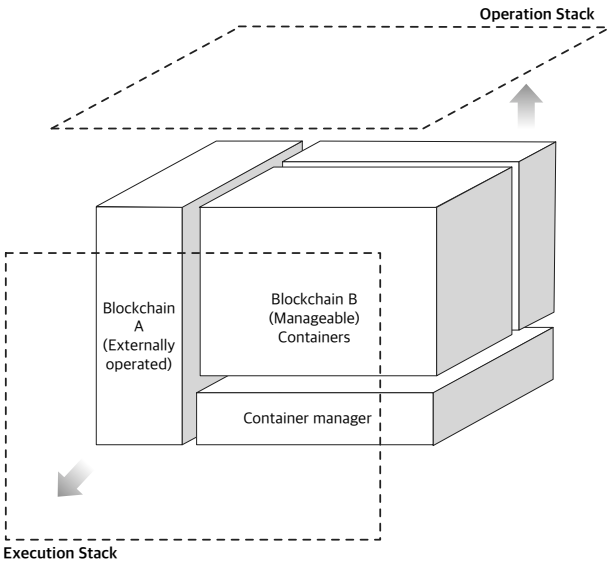

Figure 2: BIFROST Node architecture

Modular design. One of the sub-goals of BIFROST Node systems is to provide flexibility in the selection of a blockchain system. A BIFROST Node has a Container manager and Containers. The Container consists of a manageable blockchain system, Linkers for interconnection between DApp components, Input/Output (I/O) interfaces for external connection, and the connection manager to handle networking. The Container manager is in charge of the creation and management of Containers and is integrated with the Docker container system.

Stacks in two views. Figure 2 shows the logical composition of the BIFROST Node. The structure of the BIFROST Node can be represented as a 3-dimensional (3D) stack to depict how the modules are constructed and how they work for DApps. First, the execution stack shows the structure of functional modules paralleling the externally operating blockchain system. In the operation stack, we illustrate how BIFROST executes the DApp components with the resources of blockchain systems.

2.4 Execution stack

The execution stack of BIFROST is shown in Figure 3. It has a layered structure to build an interworking platform with multiple blockchain systems. Thus, the execution stack has two pillars: one for the (existing) public blockchain system and the other for the managed blockchain Container. Most linkage modules reside in the BIFROST Container.

We assume that a blockchain subsystem including a consensus algorithm and network module and a runtime environment to execute smart contract. The BIFROST Node uses the externally operating blockchain system without any modification. BIFROST runs the multiple components

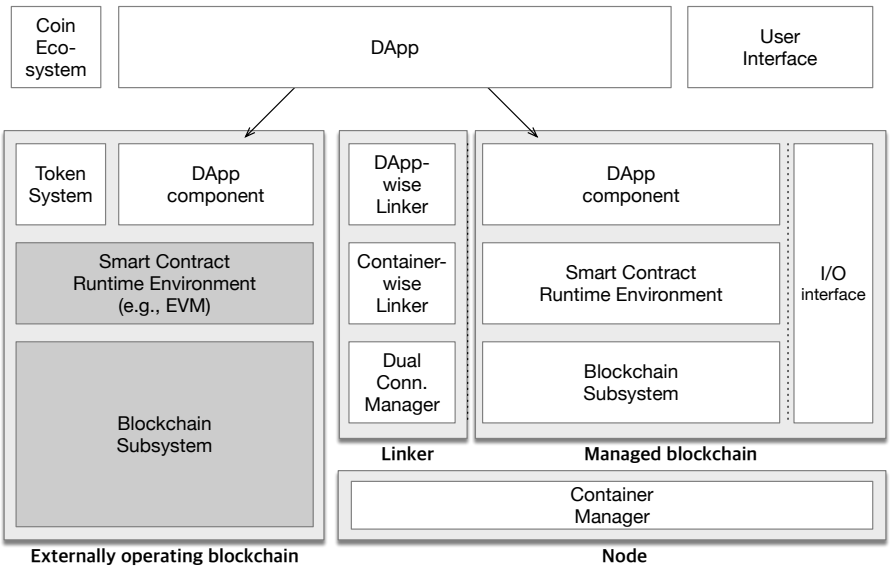

Figure 3: BIFROST execution stack

from DApps and maintains a token system of BIFROST, BFC (§4.1.1).

Container. A Container has a blockchain subsystem and the following modules:

- Blockchain Subsystem + Smart Contract Runtime Environment: Any blockchain subsystem is applicable if it has basic functionality on a smart contract.
- Connection Manager: The connection manager maintains the duplex connections among multiple blockchain systems. When the managed blockchain network forms a federation, where multiple institutes participate in a network, the connection manager also supports the connection between the Nodes in the federation.
- Linker: BIFROST splits a single DApp into components and deploys them to each target blockchain subsystems. During the split, BIFROST inserts stub codes into each component for interoperation. Linkers help the components to interwork with the stub codes. Linkers are classified as the DApp-wise and the Container-wise Linkers for interoperation at the level of DApps and Containers, respectively.
- I/O Interface and I/O Taps: The external connections for web, external storage, oracle (data feed) are supported through the I/O Interface module. In order to interact with the smart contract, the I/O Interface uses the I/O Tap, which is another smart contract account, as a connecting account.

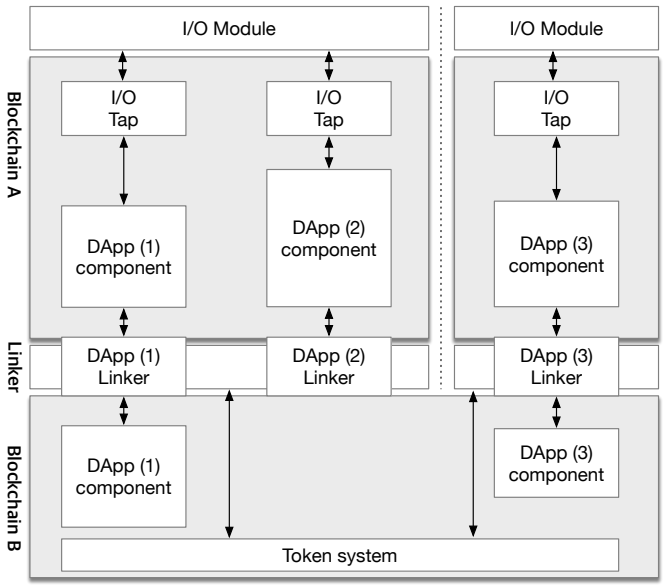

Figure 4: BIFROST operation stack

2.5 Operation stack

Figure 4 depicts the operation stack, which shows how DApp components work with other modules. Each DApp has its own a Linker and an I/O Tap. They both reside in the BIFROST Container. The configurations of the Linker and I/O Tap are generated when BIFROST splits the components since they are dependent on the internal structure of the DApp.

The behaviors of the component are obviously logged and validated by the underlying blockchain. However, the Container-wise Linker manages the cost for execution (e.g., gas in Ethereum) depending on the execution/pricing model of the blockchain. In addition, BIFROST has a token system, called "BFC." By the DApp-wise Linker converting BFC to the native coin for the component in the managed blockchains, all the components on BIFROST can have a unified token system.

I/O Interface and Tap A Container has the I/O Interface module for external connection, which interacts with the components via the I/O Tap. The I/O Interface modules helps the DApp to employ the external/decentralized storage and periodic operations requiring timers.

3 Recipe for Multi-component DApps

3.1 Motivations

BIFROST lets DApps reap the benefits from the multiple blockchain systems. To do so, the DApp development requirements can be summarized as follows:

1. The split components of a DApp can interoperate as a DApp.
2. Each component should be transformed into a native form of the underlying blockchain system.

Therefore, the approach of BIFROST is to *split and transpile* DApps for the multiple blockchain environments.

Abstraction layers for universal programs. Traditionally, to make a program run on heterogeneous computing systems, one adds an abstraction layer, such as a virtual machine or OS, with a newly defined abstract instruction set. However, in the abstraction layer approach, the abstract instructions are translated to native instructions during execution on the fly. This indirect execution not only hinders the use of machine-specific features in the program but also slows down overall performance.

Transpilation to eliminate abstraction layers. BIFROST DApps and universal programs have similar goals. We want to make DApp semi-universal on the specific combination of the multiple blockchains and to avoid the limitations of abstraction layers. Thus, we take the multi-stage programming approach, which is known for "abstraction without guilt." We represent a DApp in a high-level language form and split it into multiple components. Then, we transpile each high-level component into the code in the lower-level programming language of the target blockchain system. The smart contracts have interesting quirks: A smart contract code is significantly smaller than the codes of the general computing environment, and it is functionally self-containing. Thus, the smart contract code is suitable for the code-level transformations.

3.2 Recipe language system

With this approach, we have to deal with the following challenges:

- DApp developers need a way to write down blockchain-agnostic (or semi-universal) DApp codes.
- The multi-component DApp should be aware of the cross-execution. Even though BIFROST Linkers facilitate the simultaneous execution of the components, it is hard to perform an action that causes the cross-execution between the blockchains synchronously. The DApp needs to adapt this asynchronous execution in the program model.
- For a smart contract, we always need to consider both the correctness and the fairness of the code. Furthermore, adding asynchrony makes the correctness checks more difficult.

To overcome the challenges, we build a new program language system for BIFROST. We guide the developers to adapt the cross execution model of the multiple blockchain systems, and support the split and transpilation process by leveraging the new program language system. We

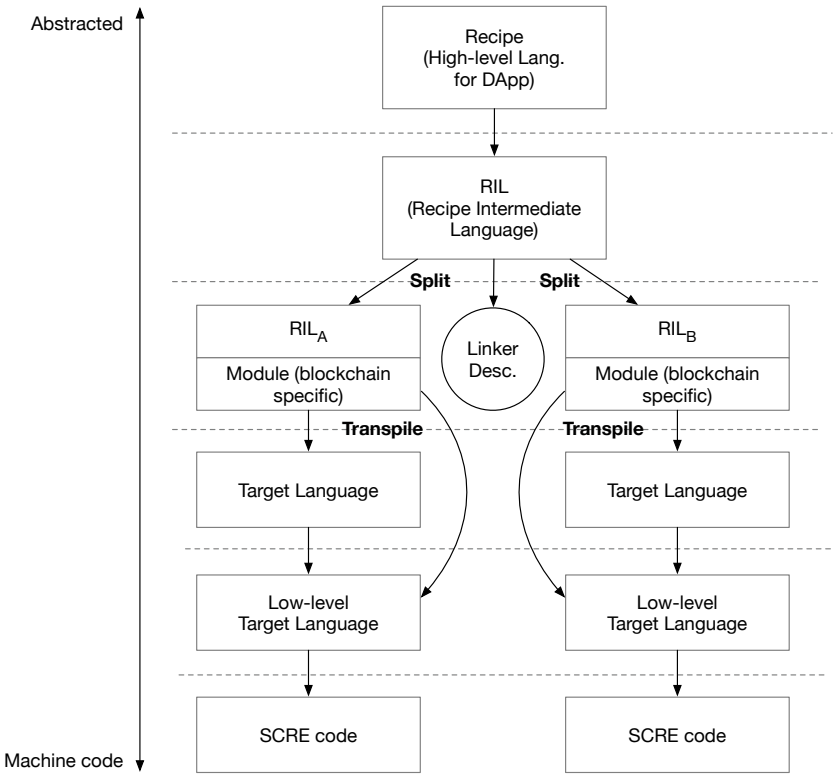

Figure 5: Recipe Language System

design a high-level language, called "Recipe," and define a Recipe intermediate language (RIL) for internal representation with transpilation rules. Figure 5 shows the structure of the Recipe language system and its stages. In Figure 5, the languages become closer to machine code (or bytecode) as they get closer to the bottom.

The Recipe Language System support the following stages:

- Recipe: A high-level, declarative programming language used to describe blockchain-independent smart contracts. It has explicit marks to designate variables and functions to a target blockchain system. It supports the asynchronous program model for the execution with the cross-state between the split components. Recipe can also append the semantic information to facilitate automatic security verification of the code.

- RIL: An intermediate language used to process and analyze the code in Recipe. The DApp code is lifted into RIL when it starts to be transformed (i.e., split and transpiled). RIL has a simple syntax for pattern-matching program analyzers. RIL also has a module system for adaptive transpilation for different blockchain systems.

- Target languages: The existing programming languages used to develop smart contracts (e.g., Solidity and Go). We classify the languages that can be used as intermediate languages

in the building process, as low-level target languages (e.g, LLL and LLVM).

- Smart Contact Runtime Environment(SCRE) code: The executable code of smart contracts in the blockchain systems.

3.3 Splitting and transpilation

Recipe supports the splitting and transpiling operations at the language levels. According to the indication in the Recipe program code, Bifrost lifts the code into RIL and divides it.

Cross variables and functions. We define the variables and the functions used in both components as "cross variables" and "cross functions," respectively. Based on the selection of the cross variables and functions, we categorize the other variables as "component local variables." Since the cross variables and functions are used between the components, the intervention of the DApp-wise Linker is required. Thus, the cross variables and the result from the call to the cross functions have two states: *transient* and *finalized*. A value of the transient state should not be used since it is under the intervention for update. Afterwards, when it has reached the finalized state, the value can be applied to the DApp. During the splitting process, Bifrost adds lock and unlock codes for each cross variable. In addition, Recipe provides a callback handler for the function call to the cross functions.

Process to build smart contract components. The splitting and transpilation process for the components is as follows:

1. Based on the marks in the Recipe code, initially classify the variables and functions for each component.
2. Lift the Recipe code into the RIL form.
3. Generate control and data dependency (super-)graphs based on the flow analyses on RIL.
4. Divide the RIL code into the components. Suppose two component as RIL_A and RIL_B.
5. Identify the component-crossing(inter-component) control flows and data flows and re-adjust RIL_A and RIL_B with the updated flow information.
6. Add stub codes on the crossing point for each RIL_A and RIL_B. Convert an assignment statement to the cross variable into a setter function, and add handlers for cross-function calls.
7. Collect the interface information on the cross variables and functions and generate the Linker description, which is delivered to Linkers to support the interoperation.
8. Automatically verify each component in terms of correctness and fairness.

9. Transpile RIL_A and RIL_B into each target language with the syntax rules for code generation and the blockchain-specific modules system.

After this process, from a DApp Recipe code, Bifrost can get the components for each blockchain subsystem and a Linker description for its DApp-wise Linker. Bifrost initiates the DApp service by deploying the SCRE codes of the components and delivering the Linker description to a Linker in the Container.

Bifrost performs automatic detecting of vulnerability patterns after generating the split components. In order to check service-wide correctness, it also applies the model-checking method with a model generated from the semantic information of the components (given by developers) and the Linker description.

4 Linker & BFC System

Sub-goals of Linkers. Bifrost combines the existing blockchain systems and lets the split components run like a single DApp. After the multi-component DApp is prepared in §3, the Bifrost Linker takes charge of the interoperation of the components. The Linker serves as a relaying proxy and it maintains shared information between the components and supports cross-function linkages. Since the smart contract of DApps has a different model from the general computing environment, the interoperation method also has a different approach. A smart contract has a permanent storage on the blockchain, a balance for native coins, as well as an internal state. Thus, the Linker has the following sub-goals to provide reliable interoperations for multi-component DApps:

- Transparent interoperation: The components from a DApp in the Recipe language should interwork via the Linker without additional work.
- Platform-wide, unified coin (token) system: The DApps in Bifrost should be supported by a unified coin (or token) system in any combinations of blockchains.
- Relay availability: Whenever the components run, the Linker is available and supports the interoperations.
- Adjustable finality: The consensus mechanism determines the characteristics on the finality of a blockchain system. For example, Proof of Work (PoW)-based blockchain systems have probabilistic finality, and Proof of Stake (PoS)-based blockchain systems generally have (pseudo) instant finality. Hence, when Bifrost combines multiple blockchain systems of different finality types, the Linker should manage the updates of different finalization conditions.

Interworking on Bifrost. In order to achieve the sub-goals, we introduce a platform-wide token system, BFC, and present the interoperation mechanism of Bifrost Linkers for the data

and control parts of DApps. For Linker availability, BIFROST provides logical and physical redundancy of Linkers executed in parallel. In addition, BIFROST helps the Linker to support adjustable finality based on the finality option in the Linker description.

4.1 Interworking via Linkers

In this subsection, we first describe a unified currency for BIFROST, called BFC, and then define the roles of Linkers in the interoperation of data and controls.

4.1.1 BFC operation

BFC resides in an existing blockchain as a token system (e.g., ERC-20 compatible tokens). Thus, the component of DApps on the blockchain can use BFC directly. For the component on the different side, BFC can be converted to the native coin if it is a managed blockchain. In that case, Linkers perform 1-to-1 exchanges of BFC to the native coins of the managed blockchain. Thus, the component can use the native coins, converted from BFC, to pay execution costs (e.g., gas) or to build coin-based services.

Note that BFC is a platform-wide token in BIFROST so that any DApp in BIFROST can use it. However, a DApp can also have its own token systems in the public blockchain system. The DApp can use all the functions of the token systems through the component on that public blockchain system.

4.1.2 States

In BIFROST, a Linker handles the following states:

- Cross variables: The permanent storage of the smart contract is differently implemented in low-level code in the blockchain systems. However, in the high-level programming languages, such as Solidity, it is abstracted as variables. Thus, Linker also identifies all cross-state information between components as variables.
- Native coin balance (managed blockchain): The balance of the component in the native coin of the managed blockchain.
- Component address: The address of the DApp component at the deployed blockchain system.
- BFC balance: The balance of BFC of each DApp.

4.1.3 Actions

The Linker also connects the followings action between components:

- Cross-function call: When a function in a component calls another function in another component, the Linker relays the function call between components.

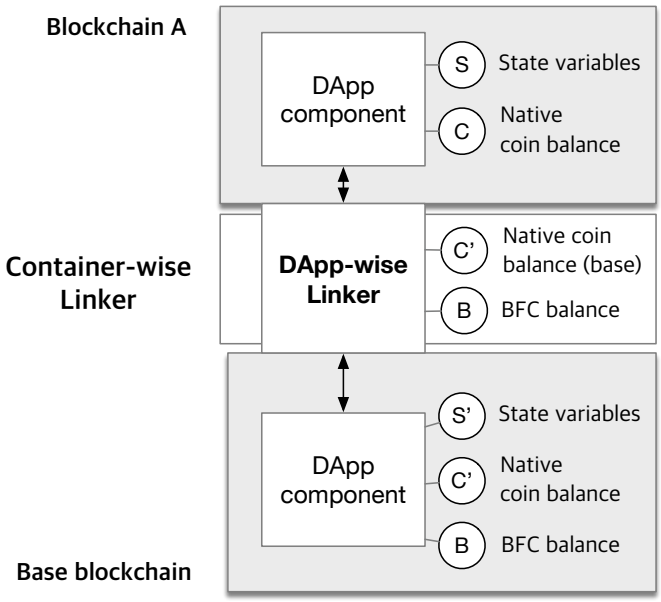

Figure 6: Internal states of DApp and Linker

- Cross-state update: When a change of a cross variable occurs, the Linker updates the variable in the other component.

4.2 DApp-wise and Container-wise Linkers

As shown in Figures 3 and 4, Linkers can be classified into two types:

- DApp-wise Linker: It is generated for each DApp and takes charge of the interoperation of the designated DApp. In the deployment phase, Bifrost configures the DApp-wise Linker based on the Linker description, which is generated during the DApp splitting process.
- Container-wise Linker: It controls DApp-wise Linkers on a Container. It works to create (replicate) and eliminate DApp-wise Linkers. The Container-wise Linker also manages the native coin balance, which is required to execute the component when DApp-wise Linkers relay cross-function calls from the components.

Figure 6 shows the internal state for Linkers and DApp components. A DApp component maintains the variables and native coin balance. Linkers do not maintain the variables for DApps but they keep track of the change of BFC. By spending the native coin balance, the DApp-wise Linker executes the smart contract.

4.3 Relaying state of cross variables

To have a consistent state on a cross variable between the components, Bifrost synchronizes the updates or transforms the access to the variable as cross-function calls. Both approaches require

the stub code addition in the Recipe splitting process and the intervention of the DApp-wise Linkers.

In the first approach, during the deployment phase, Bifrost puts an identical cross variable in both components and replaces the definition (assignment) statement with a setter-function call. The setter function marks that the cross variable is in the transient state and notifies the DApp-wise Linker of the update. The DApp-wise Linker then relays the update by calling the setter function in the other component. In the second approach, Bifrost locates the cross variable in a component side and replaces all the access to the variable in the other component with the getter and setter functions. This option reduces the process of a state update to that of cross-function call. Thus, it is more advantageous when a cross variable is mainly used in a component only.

4.4 Linker availability

A DApp-wise Linker interacts with its components through an account in each blockchain. However, the task of the DApp-wise Linker is executed on Containers. In Bifrost, multiple Containers can form a federation blockchain networks that are physically or logically separated. Whenever a Container is created, it replicates all the information including the Linker description from an existing Container. Thus, the availability of Linkers is provided by the parallel, redundant operation of Bifrost Containers, which is multiplexed via a shared account in the blockchain.

4.5 Relaying cross-function calls

The DApp-wise Linker delivers a call from a component to the cross function in the other component. In the Recipe splitting process, Bifrost replaces the cross-function call with a stub code that notifies the Linker of the call requests. If the blockchain system supports an event system, such as Ethereum, the DApp-wise Linker can be notified of the call request by subscribing to an event. Otherwise, it monitors the transactions for the address of the component in order to catch the call request.

Figure 7 shows an example of the interoperation of states and actions. The DApp-wise Linker can relay the intra-DApp cross-function call between components. To handle an inter-DApp call, where an external smart contract calls the cross function, Bifrost also adds a dummy function having the same function signature as the cross function. The dummy function also notifies the Linker of the call request.

DApp developers need to assume that the cross function can be asynchronously executed in Bifrost. Thus, the Recipe program supports callback handlers on the results of cross-function calls.

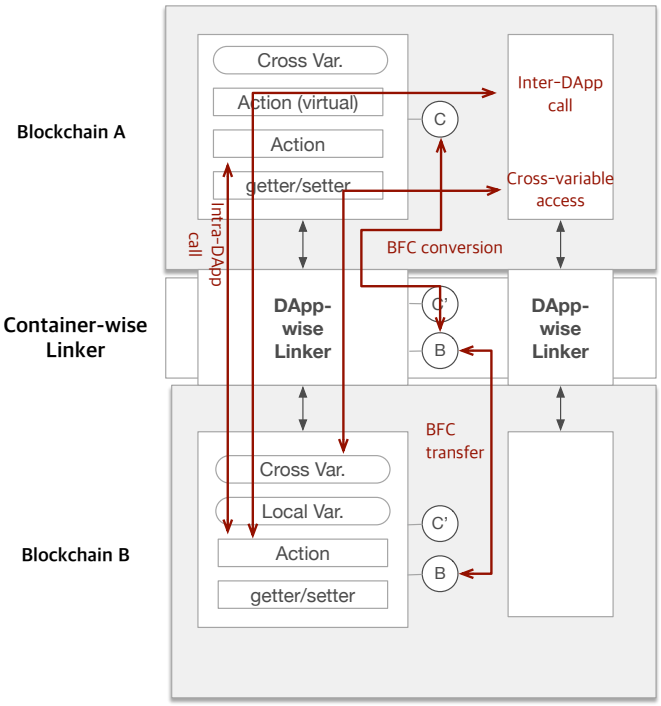

Figure 7: Interoperation of states and actions via DApp-wise Linkers

4.6 Relaying coins

As a platform-wide coin, BFC can be used everywhere on Bifrost. However, it should be converted into a suitable form to be used in DApps. When transferring an asset to an external system, several methods for atomic swap are used (e.g., proof of burn, multi-locks). Bifrost also moves BFC among blockchains. However, the conversion of Bifrost is a simpler case: the Container operator-backed DApp-wise Linker automatically converts all the received BFC into the native coin, and transfers it to the predefined address of the component. Thus, via a hashed timelock contract with the Linker, BFC can be securely delivered.

Note that the component and the Linker of a DApp have separated balances. The balance of the component is controlled by the DApp. The balance of the Linker is used only for conversion or payment for transactions (or gas).

4.7 Linker description

The Linker description is automatically generated in the Recipe splitting process. A Linker description contains the signatures of cross variables and cross functions. It also holds the handler functions for finalized cross variables and functions. In particular, to manage different finality types between blockchains, the condition for the finalized result is also appended. For example, for the component in the PoW-based blockchain, this condition tells how many confirmations

should be obtained before the result is finalized.

5 Container

5.1 Multiple Containers for service availability

In Bifrost, we consider the following conditions for building the multiple blockchain systems:

- To ensure the fundamental security and availability of the private blockchain network, the configuration of multiple Nodes is required.
- When multiple institutes participate in a blockchain network as a federation, the Nodes should be flexibly configurable to support the requirements of the federation.
- The availability of the services provided by Bifrost, such as Linkers, is as important as that of blockchain networks.

We devise the Container structure based on the idea that Bifrost modules needs availability by redundancy. Thus, we bind a managed blockchain subsystem and the supporting modules of Bifrost into a Container as shown in Figure 3. The Bifrost Containers that forms the same private blockchain network perform the same tasks simultaneously. The Container manager controls the Containers to maintain a desirable availability and performance.

5.2 Container manager

When Bifrost creates a new Container to add more Nodes to a blockchain network or to initiate a new standalone network, the Container manager handles the process. By adjusting the network configuration of Containers, the Container manager can create a federation of Containers, while each Container itself is independent. A new standalone Container manager can also be set up for isolated blockchains.

Creation by replication. When a Container is added to a federation network, the embedded blockchain system is automatically replicated due to the intrinsic characteristics of the blockchain. The Container manager additionally replicates the Bifrost modules (e.g., Linker) to the newly created Container. More specifically, it copies the Linker descriptions and Linker account information from an existing Container to the new Container.

5.3 Container/Node operator

An Operator is a service provider of a Container, and it provides the Bifrost platform service to DApp providers/developers. Since Bifrost guarantees autonomy for each Container, a third-party Container can run a Container service in a standalone manner or in a federation. Thus, a Container operator can use different combinations of blockchains with its own pricing policy for DApps.

6 Bifrost Ecosystem

6.1 Bifrost ecosystem overview

In order to increase and diversify the new DApp services on the Bifrost platform, an ecosystem where participants work with one another is needed. Bifrost aims to construct a sustainable DApp ecosystem with decentralized governance among participants and a well-defined incentive system. The main participants of Bifrost can be defined as follows:

- Bifrost Operator: Operates one or a federation of blockchains contained by one or more Containers on specific blockchains and provides services to the DApp Providers.
- DApp Provider: Chooses a Bifrost Operator and runs its own DApp

The ecosystem goals of Bifrost are as follows:

1. Bifrost Operators ask reasonable operating fees of DApp Providers, who pay them for Bifrost Platform services.
2. Bifrost Operators provide agreed service plans to DApp Providers.
3. Bifrost facilitates DApp Providers' migration among Containers to ensure optimal service for DApps. Bifrost guides the Operators to develop and upgrade their services through competition.
4. As the Bifrost ecosystem matures, all the participants benefit from the economies of scale and system sustainability.

To maintain the Bifrost ecosystem, we designed a unified currency system, BFC. BFC can be used platform-wide, considering Bifrost's nature of combining various blockchains. BFC is used for the transactions between DApp Providers and Operators and as an incentive tool for participants.

6.2 Mechanisms

Since Bifrost Operators are paid fees by DApp Providers for providing services, they are responsible for reliable system operation and fair charges. In addition, DApp Providers try to find better Operators who charge reasonable fees and provide high-quality services. In the Bifrost ecosystem, all the participants with their own goals make a fair competitive market through information asymmetry using BFC.

Main process. Operators promote their service plans by highlighting their low fees and the strength of their Containers, and DApp Providers choose optimal Operators after a careful review process. The amount of BFC staked in each Operator shows the current status of DApp Providers' choices. Since Bifrost facilitates migration among Operators, DApp Providers can choose

another Operator whenever there is any security or performance defect. Therefore, the DApp Providers' BFC staking information can be regarded as an indicator of the reliability of Operators. We call this information "Operator Credit." Since BFC resides on a public blockchain, it is easy to confirm information on the status of Operator Credit. BIFROST guarantees information symmetry by providing tools that enable this kind of information to be checked easily.

Service plan publication. BIFROST Operators promote their service plans and content as follows:

- Basic Operator information: Provide detailed information about the services through Containers and the structure of combined blockchains. (The security mechanism for other support systems for performance upgrades can be added.)
- Pricing function: Shows the fee mechanism of DApp use. Operators can charge BFC based on DApp Providers' frequency of use (complexity of transactions or use of the components) and ask for the predefined amount of BFC for staking. To define the price function, they must parameterize the complexity of transactions, the staking amount of BFC to use the service, etc. Operators have to provide pricing functions using their defined parameters.
- BFC staking per DApp: To prevent Operators' malicious behaviors, Operators must stake BFC proportional to the number of active DApps on its Containers. This is a kind of security deposit so that a DApp Provider can claim through the blame process if it does not receive an appropriate service from the Operator. DApp Providers can see how many DApps are active and the available slots left in the Container by checking the amount of BFC that the Operator stakes.

Optimal Operator selection. All the statuses of Operators in BIFROST are provided with their service plans. When a DApp Provider chooses an Operator, the BFC amount asked for in the service plan must be staked, and the actual fee is charged based on the pricing function. DApp Providers' choices can be verified by the amount of BFC that Operators get (Operator credit). Operator credit can be an indicator of Containers' reliability. Hence, Operators can demand more BFC staking for better services.

Penalty and blame. Operators can be forced to pay penalties for inappropriate service operation through DApp Providers' blame process. We designed a mechanical blame process. An Operator signs the service-level agreement according to the parameters in the service plan with a DApp Provider. If the Operator breaks the agreement, the BFC staked for the corresponding DApp is reimbursed automatically by a smart contract. As a complementary measure, a penalty based on a vote by the Operators' committee can be considered. All Operators automatically become members of the Operators' committee and have voting power proportional to their Operator credit. If a DApp Provider creates a blame report on a certain Operator to the Operators'

committee by staking BFC, the member of the Operators' committee must accept or reject it within a set period. If the blame report is accepted, the BFC staked for the corresponding DApp is reimbursed.

6.3 Bootstrapping

The Bifrost ecosystem has a network effect. That is, the entire Bifrost platform becomes more stable and efficient as the number of participants increases. Given the proportional effect of the number of Containers and the quality of the services offered, ecosystem effects are apparent not only in terms of network effects but also in terms of the type and diversity of the supported mainnet protocols. In addition, it is a "two-sided market," where DApp Providers and Bifrost Operators prosper together as the market gets bigger. Therefore, initial effort to trigger the network effect is needed. Bifrost designed the step-by-step approach as follows:

- At the initial stage, Bifrost takes the lead in Container operation: Support for pricing function and initial model which can be the basis of Container operation.
- Bifrost partner organization operate Containers as additional Operators: Manage various promotion to expand the number of Operators.
- As the Bifrost ecosystem matures, they open the Operator qualification to ensure better services under a market economy with free competition.

Bifrost's token, BFC, is initially issued with a certain amount, and there is no subsequent mining. If the DApps are attracted to the purpose of Bifrost and the ecosystem is established, the demand for fee-type BFCs provided to the Container Operators in addition to the BFCs required for the initial stage staking will also increase.

The network effect of Bifrost ensures the overall platform stability and better service delivery and also guarantees the value of BFC.

7 Conclusion

A common first question among those interested in blockchain services is what the most popular "Killer App" would be to date. The moment when anyone can answer this question in agreement will be the time when blockchain technology becomes popular. In other words, the first step that blockchain can take to change the world is the universalization of DApps where the public can experience the service.

The goal of Bifrost is to present solutions to speed and scalability in order to solve the current DApp development problems and to support the practical activation of its ecosystem. However, we do not follow the way many protocols are going through and instead use the existing system. Multiple blockchain systems currently operate in many fields and each have their own strengths. We aimed to address the problems in the DApp development environment by providing

the unique strengths of the different blockchains simultaneously through Bifrost. The Bifrost platform will grow with the advancement in blockchain technology, and DApps can adopt new technologies under stable circumstances without changing platforms and maintain.

Bifrost is innovative because it provides the most efficient solutions to "the platform risk" problems of DApp development environments. Bifrost has the potential to expand the horizons of DApps significantly, and will ultimately create an opportunity for the blockchain technology to be actively used by the public.