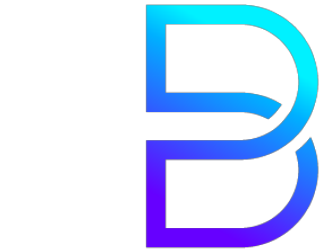

# BIFROST

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.

# 차 례


**1 Introduction** **4**

1.1 Motivations and Problems . . . . . 4

1.2 Approaches for Efficient Solutions . . . . . 5

1.3 Bifrost Method . . . . . 5

**2 Bifrost System** **6**

2.1 Design . . . . . 6

2.2 Overview . . . . . 8

2.3 Node Architecture . . . . . 10

2.4 Execution Stack . . . . . 11

2.5 Operation Stack . . . . . 12

**3 Recipe for Dual-component DApps** **13**

3.1 Motivations . . . . . 13

3.2 Recipe Language System . . . . . 14

3.3 Splitting and Transpilation . . . . . 16

**4 Linker & BFC System** **17**

4.1 Interworking via Linkers . . . . . 18

4.1.1 BFC Operation . . . . . 18

4.1.2 States . . . . . 18

4.1.3 Actions . . . . . 19

4.2 DApp-wise and Container-wise Linkers . . . . . 19

4.3 Relaying State of Cross Variables . . . . . 20

4.4 Linker Availability . . . . . 20

4.5 Relaying Cross-function Calls . . . . . 21

4.6 Relaying Coins . . . . . 21

4.7 Linker Description . . . . . 22

**5 Container** **22**

5.1 Multiple Containers for Service Availability . . . . . 22

5.2 Container Manager . . . . . 23

5.3 Operator . . . . . 23

**6 Bifrost Ecosystem** **24**

6.1 Bifrost Ecosystem Overview . . . . . 24

6.2 Mechanisms . . . . . 24

6.3 Bootstrapping . . . . . 26

**7 Conclusion** **27**

# 1 Introduction

## 1.1 Motivations and Problems

블록체인이 가져올 미래의 혁신에 대하여 많은 사람들이 긍정적인 평가와 함께 큰 기대를 하고 있다. 비트코인은 탈중앙화를 통한 화폐 기능의 가능성을 보여주었고, 이더리움은 스마트컨트랙트를 이용하여 기술적인 진보와 함께 수많은 어플리케이션들이 실질적으로 동작할 수 있는 기반을 만들었다. 스마트컨트랙트와 이를 활용한 Decentralized Application (이하 DApp)은 블록체인위에서 여러가지 서비스를 가능하게 한다. 특히 다양한 DApp의 등장은 블록체인 생태계를 활성화시키며 DApp의 우월한 인센티브 시스템과 분산화를 통한 투명하고 안정적인 시스템의 구조는 가까운 미래에 소프트웨어 산업 전체의 큰 틀을 바꾸는 역할을 할 것이라 예상된다.

하지만, DApp의 발전속도가 대중의 기대에 크게 미치지 못하고 있으며, 블록체인 기술 자체의 유효성에 대한 의구심마저 제기되기도 한다. 특히, 1,000개가 넘는 DApp을 보유한 이더리움 네트워크가 속도의 문제로 유효한 기술적 지원을 하지 못하고 있다는 점에서 그 한계가 제시되기도 한다. 또한, 소위 Trilemma로 불리는 Decentralization, Safety 그리고 Scalability를 모두 프로토콜 레벨에서 해결하려는 노력이 DApp 보다 프로토콜의 수가 더 많아지는 "Protocol Fever"의 현상을 가져왔다. DApp의 개발자 입장에서 가장 먼저 결정해야 할 일이 바로 DApp 서비스를 제공할 메인넷 프로토콜 선정이라는 사실로 비추어 볼 때, 현재의 Protocol Fever 현상은 안정적인 DApp 개발에 커다란 장애가 될 수 밖에 없다. 다양한 프로토콜과 다양한 DApp의 존재는 서로의 수요에 대한 미스매치현상을 심화시킨다.

이러한 블록체인 기술의 구조적 문제점들에도 불구하고 국가와 산업을 초월하여 블록체인은 혁신을 가져오고 있으며, 대부분 프라이빗 블록체인을 이용하여 상용화를 이루고 있다. 특히, 저작권 및 물류 분야에서 프라이빗 블록체인은 고유의 데이터 신뢰도와 성능 개선을 통해 새로운 분야를 개척하고 있다. 예를 들어, 2018년 10월 상용화된 블록체인 기반 식품 추적 네트워크 "IBM Food Trust"는 식품 원산지부터 운송 기록 등 모든 세부 사항을 공유해 블록체인을 활용한 시스템이다.

블록체인이 혁신적인 서비스로 자리잡기 위해서는 역동적인 DApp의 생태계가 무엇보다도 필요하다. 모든 첨단기술의 초창기와 마찬가지로, DApp이 서비스의 형태로 자리잡고 생태계를 구성하여 선순환의 구조가 형성되기 위해서는 극복해야할 문제들이 여러가지 있다.

우리는 이 장에서, 앞에서 기술한 블록체인의 기술적 환경을 바탕으로 DApp의 개발 환경에서 생길 수 있는 문제점들을 정의하고 그에 대한 해결책의 전제 조건을 제시하며 결론적으로 DApp들이 작동할 수 있는 Bifrost 솔루션을 제시한다.

DApp의 개발환경에서 눈에 띄는 세가지 문제점들이 발견된다.

첫째, 플랫폼 리스크이다. 메인넷 프로토콜 위에서 서비스를 제공한다는 관점에서 DApp은 이더리움이나 이오스 같은 블록체인 플렛폼을 선택해야 한다. 수많은 메인넷 블록체인이 개발되었으며 지금 이순간에도 필요한 기능을 강조하는 여러가지 메인넷들이 발표되고 있다. 현재 존재하는 모든 메인넷 프로토콜은 모두 서로 다른 장단점을 보유하고 있지만 그 어떤 것도 DApp들에게 완벽한 개발환경을 제공하고 있지않다. 그럼에도 불구하고 DApp의 개발자들은 메인넷을 선택해야하고

결국 그 메인넷 프로토콜 시스템의 확장성 및 개발 방향에 따른 제어할 수 없는 리스크를 갖게 된다.

둘째, DApp의 메인넷에 대한 요구사항이 너무 다양하다. 목적에 부합하는 원활한 서비스를 위한 시스템 요구가 DApp마다 다를 수 있다. 하지만, 다양한 요구를 모두 수용할 수 있는 하나의 메인넷 프로토콜은 존재하지 않는다. 예를 들어, 블록체인 시스템을 이용하는 크립토게임은, 수많은 트랜잭션의 처리가 빠르게 수행되어야 하며 안전한 리워드 시스템이 필요하다. 즉, 속도에 대한 확장성과 안전성을 갖는 분산화된 환경이 동시에 필요한 것이다. 이러한 요구사항을 하나의 퍼블릭 블록체인에서 모두 만족시키는 것은 거의 불가능하다.

셋째, 프라이빗 블록체인을 통한 문제해결에 한계가 있다. 퍼블릭 블록체인의 확장성에 대한 문제를 부분적으로 또는 전체적으로 프라이빗 블록체인을 통해 해결하려는 많은 연구가 진행중이다. 하지만 프라이빗 블록체인은 그 안정성 검증시스템이나 믿을 만한 코인생태계를 구성하는 점에서 퍼블릭 블록체인에 비해 어려움을 가지고 있다.

## 1.2 Approaches for Efficient Solutions

우리는 DApp의 환경에서 생기는 세 가지 문제점에 대하여 실질적인 해결 위한 다음의 세가지 접근 방법을 고려한다.

- 첫째, 하나의 블록체인 메인넷이 모든 상황의 완벽한 해답이 될 수 없다. 완벽한 블록체인이 존재하지 않고 서로의 장단점이 다른 상황에서 하나의 메인넷 블록체인이 플랫폼으로 고착화되어 사용하는 것은 비효율적이고 위험하다.
- 둘째, 프라이빗 블록체인이 모든 것을 해결할 수는 없다. 프라이빗 블록체인은 많은 장점을 가지고 있지만 퍼블릭 블록체인의 본질적인 장점을 대체할 수는 없다.
- 셋째, OS나 Abstraction Layer의 추가를 통한 해결은 전체적인 오버헤드만 늘릴뿐이다. 메인넷의 확장을 통한 기능의 개선은 전체 시스템의 과부하에 대한 요인일 수 있으며, 확장성에 대한 제한을 가져올 수 있다.

## 1.3 Bifrost Method

우리는 Bifrost를 통해서 퍼블릭 블록체인의 안전성과 프라이빗 블록체인의 성능을 가진 DApp 플랫폼을 가능하게하고자 한다. 우리는 당면한 문제들을 해결하고 블록체인의 실제 DApp 서비스에 집중하기 위해서, 기존의 블록체인 시스템들을 Building Block으로 사용하여 개별 시스템들의 장점을 취하면서 독립적으로 서비스를 구성할 수 있는 새로운 플랫폼이 필요하다는 결론에 이르게 되었다. 즉, 검증신뢰도를 바탕으로 하는 퍼블릭 블록체인과 목적에 맞게 튜닝된 프라이빗 블록체인을 조합하여 하나의 실행환경으로 구성하고 그 위에 각각의 장점을 활용할 수 있는 DApp 플랫폼을 개발한다면 앞에서 제기한 모든 문제들과 그것의 해결책에 대한 방향에 부합할 수 있다.

우리는 DApp을 위한 새로운 플랫폼, Bifrost를 소개한다. Bifrost는 실제 블록체인의 서비스가 동작하는 DApp에 초점을 맞춘다. Bifrost의 기반 플랫폼 블록체인 시스템은 검증된 기존

의 블록체인 시스템을 퍼블릭 또는 프라이빗 블록체인 파트로 바로 활용하고, 조합된 환경 위에서 DApp이 동작할 수 있도록 DApp을 재구성(restructure)하여 실행시킨다. 이를 위하여 Bifrost는 DApp의 deployment 단계에서 operation 단계까지의 전과정을 지원한다. Deployment 단계에서는 DApp 개발자의 의도를 반영하여 DApp의 코드를 퍼블릭 파트와 프라이빗 파트로 분리하며 자동으로 재구성한다. 재구성 과정에서 생성된 컴포넌트들이 각각의 위치에 설치되고 나면, DApp의 operation 단계에서는 컴포넌트들이 안전하게 연동하여 실행될 수 있도록 Bifrost가 연결점(Linker) 역할을 한다.

Bifrost는 예측하기 힘든 블록체인의 미래에 대비할 수 있는 유연성을 가지고 있다. Bifrost에서는 다음과 같은 특징들을 가지고 있다.

1. **Blockchain as a Building Block:** 퍼블릭 블록체인과 프라이빗 블록체인을 함께 묶어 하나의 병렬환경을 구성한다. 이때 프라이빗 블록체인은 Federation 형태로 여러 개의 체인을 함께 구성할 수 있다.

2. **Container 단계에서의 자유도:** 하나의 프라이빗 블록체인을 독립된 Container 형태로 구성하고, 각 Container마다 다른 설정이 가능하다. 즉, Container는 동적으로 생성과 삭제가 가능하고 내부적으로 성능튜닝이나 트랜잭션의 익명성을 보장하는 기술들을 적용할 수 있다. Container 레벨에서의 관리는 기존의 시스템에서 고려하지 못한 접근 방법이 가능하다. 예를 들어, 기록을 남기지 말아야 하는 서비스는 독립된 Container로 운영 후 파기할 수 있다.

3. **Dual-component DApp 재구성:** Bifrost에서 개발된 DApp은 기반 블록체인에 맞는 스마트 컨트렉트의 컴포넌트들로 재구성되어서 퍼블릭과 프라이빗 블록체인 위에서 동시에 동작한다. 따라서, 프라이빗 블록체인의 높은 성능을 구현하면서도 퍼블릭 블록체인을 통해 안전하게 검증을 받을 수 있다. Dual-component DApp의 재구성 기술은 초기에 기존의 DApp을 대상으로 하지만 Bifrost의 유연성을 모두 지원하기 위한 새로운 언어를 통해서 확장될 예정이다.

Bifrost의 진정한 가능성은 서로 다른 기존의 블록체인 시스템을 장점을 살리며 조합할 수 있다는 점이며, DApp이 궁극적으로 플랫폼 리스크에서 자유롭게 할 수 있다는 데에 있다.

# 2 Bifrost System

## 2.1 Design

**DApp's dilemma or trilemma.** 서로 다른 특색을 가진 블록체인 시스템이 시장에 선보이면서, DApp은 목적에 따라서 자신에게 가장 적합한 장점을 가지고 있는 블록체인 시스템을 선택할 수 있게 되었다. 하지만 DApp 서비스가 하나의 블록체인 시스템에 종속되면서 해당 블록체인 시스템이 가지는 단점 또한 감내해야 하기 때문에 이러한 선택이 DApp 개발자나 서비스 제공자 입장에서는 쉽지 않은 상황이다. 예를들어, digital contents를 판매하는 DApp에게는 실제 비용이 오고가는 입금이나 거래 공지 등의 기능은 퍼블릭 블록체인이 적합하지만, 내부적인 경매나 협상 기능 등은 프라이빗 블

록체인이 더 적합하다. DApp 제공자는 이러한 dilemma (또는 trilemma) 속에서 서비스의 최종적인 안전성과 성능을 결정하는 블록체인을 한번에 정확하게 선택해야 하는 어려움에 직면해 있다.

**Don't reinvent the wheel.** 기존의 블록체인의 한계점을 해결하기 위하여 이미 많은 새로운 블록체인 시스템들이 활발하게 제안되고 개발되고 있지만, 대부분 불록체인의 시스템 제공자 (provider) 측면에서 전혀 새로운 블록체인 시스템과 환경을 구축함으로써 모든 문제를 한꺼번에 해결하려는 접근을 하고 있다. 하지만 하나의 답이 DApp 마다 다른 세부적인 요구사항들을 모두 충족시키기 어렵고, 근본적이고 기술적인 trade off들 때문에 하나의 해결책이 결국 또다른 문제점을 불러오는 악순환이 반복되고 있다.

**Our approach: cherry-picking in a good way.** 우리는 기존의 블록체인 시스템으로 구성된 퍼블릭과 프라이빗 블록체인이 조합된 이중 블록체인(dual blockchain) 환경을 마련하고, DApp이 두 블록체인 시스템의 장점을 이용하며 실행될 수 있도록 하는 플랫폼을 제공한다. 우리는 Bifrost라는 새로운 DApp 플랫폼을 통하여 DApp에 대한 실용적인 해결책을 제공하고자 한다. DApp 서비스의 개발자와 제공자 입장에서는 1) 기반 블록체인 시스템에 종속되는 과정에서 오는 불확실성과 2) 하나의 DApp에서 퍼블릭과 프라이빗 이라는 두 가지 블록체인 시스템의 특징을 동시에 필요로 하는 역설적 요구조건들을 해결해야 한다. 결국 DApp에서 가장 중요한 것은 어떠한 블록체인을 사용하느냐 보다는 플랫폼에서 DApp의 요구사항들을 잘 해결해 줄 수 있는가 있다. 따라서 우리는 새로운 mainnet 블록체인 시스템을 만들기 보다는 어떻게 DApp이 기존의 블록체인 시스템에서 필요한 기능들을 사용할 수 있게 할지에 집중한다.

Bifrost는 DApp의 개발에서 실행까지의 전 과정을 지원하는 플랫폼이다. Bifrost는 DApp 서비스 개발자가 서로 다른 장점을 가진 블록체인 시스템이 조합된 환경을 가정하여 서비스를 개발하고, 설치 단계에서 자동으로 DApp smart contract를 재구성하여 운영할 수 있도록 한다. 일반적으로 하나의 DApp 서비스는 핵심이 되는 smart contract 코드들을 기반으로 추가적으로 web이나 storage와 같은 외부 인터페이스를 통하여 사용자에게 서비스 되는 형태를 가지고 있다. Smart contract는 블록체인 시스템마다 차이가 있지만, 개발과정에서는 다양한 기능들을 모듈화한 함수들과 정보들이 저장되어 있는 변수 또는 state의 조합으로 이루어져 있다. 따라서 DApp의 각 기능은 smart contract의 함수 단위로 구분하여 생각할 수 있다. DApp 개발자는 기존과 동일한 과정을 통해서 DApp 서비스를 개발하면서 각 기능별로 대상이 되는 블록체인 시스템을 지정하며 명시한다. 이후 설치 단계에서 Bifrost는 DApp smart contract의 각 기능을 기능별로 적합한 블록체인 시스템에 매핑(mapping)하고, smart contract를 분리하여 선택된 블록체인 시스템에 맞도록 조정하는 재구성 과정을 거친다. 하지만, 이러한 과정에서는 정교한 smart contract에 대한 분석과 분리된 부분들이 seamless하게 연동되기 위한 플랫폼의 지원이 필요하다. 따라서 Bifrost는 DApp의 재구성과정에서 파트간 연동에 필요한 정보를 자동을 생성한다. 실제 운영단계에서 Bifrost는 분리된 DApp의 조각들이 개발자의 의도대로 하나의 DApp으로 실행될 수 있도록 하는 다리의 역할과 외부 연동을 위한 DApp 플랫폼의 역할을 수행한다.

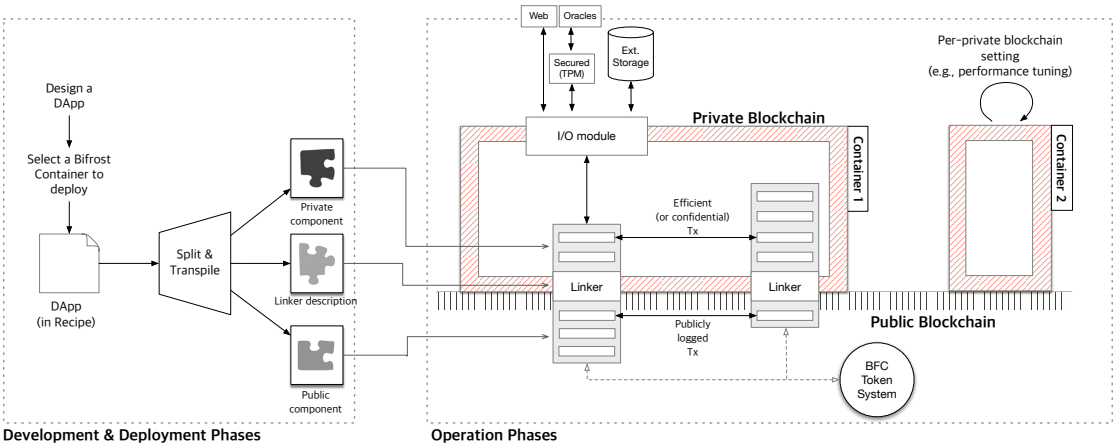


그림 1: BIFROST process

**Design goals.** BIFROST의 설계 목표는 다음과 같다.

- DApp이 하나의 블록체인 시스템에 종속되지 않도록 한다.
- DApp을 위하여 기존의 블록체인 시스템을 조합한 이중 블록체인(dual blockchain)의 환경을 제공한다.
- DApp이 이중 블록체인에 걸쳐서 동작하여 각 블록체인의 장점을 활용할 수 있도록 한다.
- 이중 블록체인에 걸쳐서 동작하는 DApp의 추가적인 성능 저하를 최소화한다. (하나의 DApp이 다른 블록체인 시스템에 걸쳐서 동작하지만, 각 부분은 하위 블록체인이 제공하는 native smart contract를 위한 runtime의 성능을 보장한다.)
- 기존의 smart contract 언어를 확장한 새로운 프로그래밍 언어를 정의하여 이중 블록체인을 위한 프로그램 모델을 제공한다.

## 2.2 Overview

BIFROST는 이중 블록체인의 장점을 살린 효과적인 DApp 플랫폼을 제공한다. 특히 §2.1의 목표를 달성하기 위해서, BIFROST는 DApp의 생애주기(lifecycle)에서 동작한다. 즉, DApp의 개발과정에서 설치 및 운영 과정을 관리한다. 전통적으로는 이기종의 복합 시스템을 통합하는 과정에서는 별도의 추상화 계층을 추가하여 하부 시스템의 복잡도를 낮추는 방법을 적용해 왔다. 하지만 이러한 경우에는 간접적인 실행과정에서의 성능저하와 함께 하부 시스템의 기능을 제한적으로 사용해야하는 문제점을 가지고 있다. BIFROST는 단순한 시스템의 확장보다는 진보된 programming language 기술을 적용하여 근본적인 해결책을 제공한다. BIFROST의 DApp은 개발 과정에서 최소한의 노력으로 이중 블록체인 환경을 고려하고, 설치 단계에서는 자동으로 다중 블록체인에 적합한 형태로 재구성된다. 재구성의 과정에서는 각각의 블록체인에서 smart contract를 표현하기 위해서 사용하는 프로그래밍

언어나 중간 언어 형태로 변환(transpile)됨으로써, overhead를 유발하는 추상화 계층 없이 실행될 수 있도록 한다. 특히 Bifrost는 프라이빗 블록체인과 퍼블릭 블록체인을 조합한 환경을 제공할 수 있어, 프라이빗 블록체인의 성능을 유지하면서 퍼블릭 블록체인에서의 토큰 시스템과 연동하거나 안전성에 대한 검증을 강화할 수 있도록 한다. 이러한 과정은 Solidity와 같은 기존의 스마트 컨트랙트 언어를 확장하는 방식도 가능하지만, 이중 블록체인을 활용하기 위한 새로운 프로그램 모델을 지원하기 위하여 우리는 새로운 DApp 프로그래밍 언어인 Recipe를 제공한다.

**How Bifrost works.** Bifrost는 이중 블록체인 시스템 위에서 DApp을 동작시키기 위하여 DApp을 재구성하고 확장한다. 이중 블록체인 환경은 Bifrost Container 들에 의해서 제공되는데, 복수의 Bifrost Container들이 하나의 프라이빗/연합 블록체인 네트워크를 구성하며 동일한 퍼블릭 블록체인 네트워크에 연결한다. Bifrost는 DApp의 관점에서 크게 (1) 개발단계(development phase), (2) 설치단계(deployment phase)와 (3) 운영단계(operation phase)로 나누어 진행된다. DApp을 개발하는 단계에서는 자신의 목적에 맞는 블록체인 시스템의 조합을 파악하고, smart contract에서 어떠한 변수와 함수가 각 블록체인 시스템에 해당하는 지를 명시한다. Bifrost는 설치단계에서 DApp에서 주어진 정보에 따라 DApp을 함수단위로 분리하고 동시에 연결에 필요한 정보를 생성한다. 분리된 DApp은 각 블록체인 시스템에 맞는 smart contract 코드로 변환된다. 그리고 재구성 과정에서 발생한 코드의 변경에 의해서 야기된 보안의 문제점과 분리된 동작에서 발생할 수 있는 보안의 문제점을 확인하기 위하여 각 부분에 대한 자동 보안 점검을 실시한다. 안전성이 확인된 DApp은 개발자가 선택한 Bifrost Container와 퍼블릭 블록체인에 설치된다. 이어지는 운영단계에서 Bifrost Container는 DApp의 분리된 부분들의 연동을 지원한다.

**The overall process.** Bifrost의 플랫폼을 이용한 DApp 서비스 과정은 그림 1과 같이 진행된다. 세부적으로는 다음과 같은 과정을 거치게 된다.

1. 개발자는 블록체인 서비스 DApp을 설계하면서 필요한 이중 블록체인의 조합을 갖춘 Bifrost Container를 선택한다.
2. Recipe 언어를 이용하여 DApp을 개발한다.
3. 개발된 DApp에 성능과 보안의 문제를 고려하여 퍼블릭 파트와 프라이빗 파트에서 동작할 기능을 표기한다.
4. DApp은 목적에 따라 프라이빗 블록체인 container에 설치될 컴포넌트와 퍼블릭 블록체인에 설치될 컴포넌트, 그리고 둘 사이를 연결하기 위한 정보를 담고 있는 Linker description 부분으로 분리되어 재구성된다.
5. Bifrost는 생성된 각 컴포넌트들은 용도에 맞추어 각자의 위치에 설치된다.

DApp의 operation 단계에서는 다음과 같은 Bifrost의 지원을 통해서 서비스를 운영한다.

- 프라이빗 블록체인 컴포넌트는 자신이 설치된 Container의 프라이빗 블록체인의 내부 기능을 이용한다.
- Bifrost Linker는 분리되어 있는 두 컴포넌트 사이의 state 정보를 일치시키고, 제어 흐름을 연결한다.
- 퍼블릭 블록체인 컴포넌트는 기존의 블록체인에서 제공하는 기능을 활용한다.

Bifrost는 새로운 블록체인을 만들지 않고 기존의 블록체인을 사용한다. 퍼블릭 블록체인과 프라이빗 블록체인을 위한 공간을 만들어 놓고 기존의 블록체인들을 각각의 위치에 조합하여 사용한다. 퍼블릭 블록체인의 경우에는 동작 중인 블록체인 시스템에 연결하고, 프라이빗 블록체인과 나머지 기능들은 Bifrost Node 시스템 위에서 운영된다.

## 2.3 Node Architecture

Bifrost의 Node 시스템은 퍼블릭 블록체인과 프라이빗 블록체인에 걸쳐 DApp이 실행될 수 있는 환경을 제공한다. Bifrost의 운영단계에서는 이중 블록체인의 장점을 활용하기 위해서 독립된 블록체인 시스템을 묶고 연결하는 기반 시스템이 필요하다. 특히 이러한 시스템은 재구성된 DApp들의 연동을 고려하여 다양한 레벨에서의 연동을 지원할 수 있어야 한다. 따라서 Bifrost Node는 stack 구조를 통하여 블록체인간 연동에서부터 DApp 파트간 연동까지 다양한 레벨의 연동을 주요 목적으로 한다.

**Modular design.** Bifrost Node는 블록체인 선택의 유연성을 보장하기 위해서 퍼블릭 블록체인을 제외한 부분을 모듈화하여 구성한다. 크게 하나의 Node는 하나의 Container manager와 그 위에서 동작하는 Container들로 구성된다. 하나의 Container는 독립된 컨테이너 구조로서 Bifrost의 프라이빗 블록체인 파트를 담당한다. Container에는 중심이 되는 프라이빗 블록체인 시스템과 함께, 블록체인과 DApp을 연결하기 위한 Linker 시스템, 외부와의 연결을 위한 I/O interface, 네트워크 연결을 담당하는 Dual connection manager 등으로 구성되어 있다. Container manager는 docker의 컨테이너 관리 시스템을 통해서 Container들의 생성과 확장 및 관리를 담당한다.

**Stacks in two views.** 그림 2는 개념적인 형태의 Bifrost Node의 구성을 보여준다. Bifrost의 Node 구조는 시스템의 구성과 및 실행의 단계를 모두 포함하고 있기 때문에, 전체적으로는 3차원의 stack 형태로 나타날 수 있다. Execution stack에서는 Bifrost의 Node가 퍼블릭 블록체인 시스템과 병행하여 하드웨어에서부터 어떠한 기능적 계층으로 구성되어 있는지를 보여준다. Operation stack에서는 DApp이 설치된 이후에 Bifrost를 통해서 퍼블릭 블록체인과 프라이빗 블록체인의 자원을 이용하여 DApp의 component들이 연동하며 실행되는지를 보여준다.

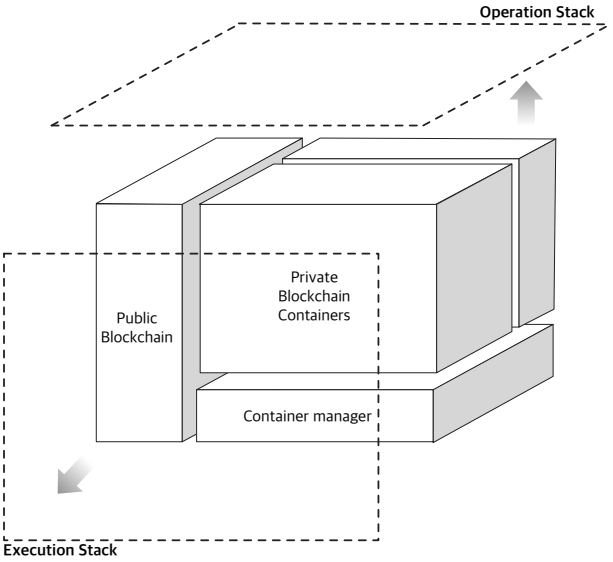


그림 2: Bifrost Node architecture

## 2.4 Execution Stack

Execution stack은 그림 3과 같이 구성된다. Execution stack은 기능적인 계층구조로서 블록체인 시스템으로부터 플랫폼을 구축하기 위해 필요한 요소를 갖추고 있다. Execution stack의 한 축은 퍼블릭 블록체인이고 나머지는 한 축은 프라이빗 블록체인을 바탕으로 Bifrost의 연동 기능들이 탑재되어 있는 Container로 이루어져 있다.

**Public blockchain.** 우리는 퍼블릭 블록체인 시스템이 합의 알고리즘과 네트워크로 구성된 Blockchain Subsystem 부분과 기본적 smart contract를 실행할 수 있는 최소한의 실행(runtime) 환경을 가지고 있다는 것을 가정한다. 퍼블릭 블록체인은 기존의 시스템을 변경없이 Node에서 해당 퍼블릭 블록체인으로의 연결을 유지하여 연동한다. 퍼블릭 블록체인에는 DApp의 public component가 실행되며, Bifrost에서 사용하는 token system인 BFC를 운영한다. (§4.1.1)

**Container.** 하나의 Container는 프라이빗 블록체인 시스템을 바탕으로 다음과 같은 주요 모듈들을 포함하고 있다.

- Private Blockchain + Smart Contract Runtime Environment: Container의 핵심은 프라이빗 블록체인과 smart contact 실행을 위한 런타임 환경이다. 기본적인 조건을 만족시키는 어떠한 프라이빗 블록체인도 적용 가능하다. 이 프라이빗 블록체인은 Container가 연결되어 있는 퍼블릭 블록체인과 같은 시스템이어도 되고, 용도에 따라 다른 설정을 가지거나 서로 상이한 시스템으로 구성할 수 있다.
- Dual Connection Manager: 블록체인 시스템 간의 네트워크 연결을 담당한다. 기본적으로는

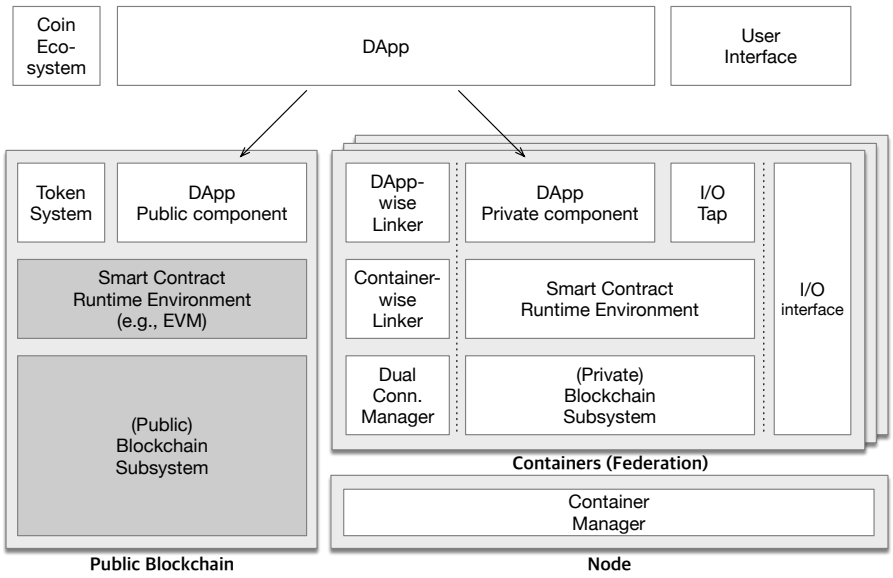


그림 3: Bifrost execution stack

퍼블릭 블록체인과 프라이빗 블록체인 양방향으로의 연결을 지원한다. 프라이빗 블록체인에서 다수의 기관이 다수의 Node로 참여하는 연합(federation)을 구성할 경우 Dual Connection Manager가 연합 노드들간의 연결도 담당한다.

- Linker: Bifrost는 하나의 DApp을 private component와 public component로 나누어 Container와 퍼블릭 블록체인에 설치한다. 이 때, 둘 사이의 연결을 위하여 각 컴포넌트에는 stub code가 생성되고 이 stub code와 Linker가 연결되어 두 컴포넌트를 연동한다. Linker는 각 DApp에 맞는 기능을 지원하기 위한 DApp-wise Linker와 이러한 DApp-wise Linker들을 전체적으로 관리하며 Container 단계에서의 연결을 담당하는 Container-wise Linker로 구분된다.
- I/O Interface와 I/O Tap: I/O interface는 web, storage, external oracle (data feed) 등의 외부 연결을 통일된 형태로 사용할 수 있도록 지원한다. I/O Tap은 프라이빗 블록체인 내에서 정의되는 smart contract로서 DApp의 private component에서 I/O Interface를 사용하기 위한 연결점으로 동작한다.

## 2.5 Operation Stack

**Private component.** 그림 4의 operation stack에서는 DApp이 운영되는 과정에서 필요한 요소를 보여준다. 하나의 DApp은 private component를 중심으로 퍼블릭 블록체인에서 수행되어야 하는 작업을 public component로 분리하여 운영한다. 각 DApp은 연결을 위한 Linker와 외부연결을 위한 I/O Tap을 별도로 가지게 된다. DApp마다의 내부적인 프로그램 구조(함수 구조)에 따라서 두 private과 public component의 연결 방식과 입출력 방법이 달라지기 때문에, Bifrost Builder에서 분리 생성하는 과정에서 Linker 설정 및 I/O Tap도 함께 생성한다.

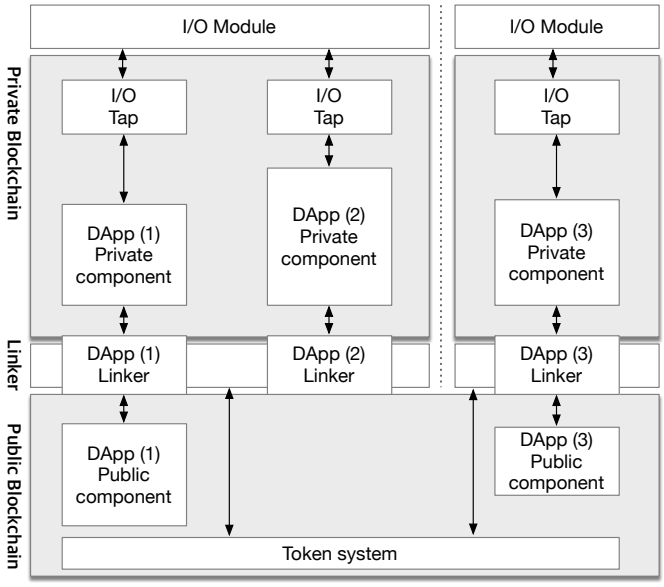


그림 4: BIFROST operation stack

**Public component.** 퍼블릭 블록체인에서 동작하는 public component는 모든 동작 과정이 안전하게 퍼블릭 블록체인에 의해서 검증을 받는다. 다만, 기반이 되는 퍼블릭 블록체인의 동작 방식에 따라서 실행을 위한 비용(e.g, gas in Ethereum)들을 고려하여 동작 방법을 결정한다. 또한, 퍼블릭 블록체인에는 BIFROST를 위한 BFC Token System을 설치하고 Linker에 직접적으로 연결되어 있어서, BIFROST위의 모든 DApp은 BFC를 안전하게 사용할 수 있도록 한다.

**I/O Interface and Tap** 하나의 Container는 Linker와 함께 외부 연결을 위한 I/O Interface를 가지고 있으며, 이는 I/O Tap과 연결되어 프라이빗 블록체인 내부로의 전달 및 전파를 담당한다. DApp의 운영과정에서 주기적으로 호출되어야 하는 기능(timer)이나 대용량 데이터의 저장 또는 검색과 같은 작업들을 I/O Tap의 smart contract에 정의되어 있는 함수를 호출하여 사용하게 된다.

# 3 Recipe for Dual-component DApps

## 3.1 Motivations

BIFROST는 DApp이 이중 블록체인의 장점을 활용할 수 있도록 한다. 그러기 위해서 DApp의 개발 과정의 요구사항은 다음과 같이 정리할 수 있다.

1. DApp이 분리된 상태에서도 하나의 DApp처럼 연동하여 동작할 수 있어야 한다.
2. 각각의 분리된 부분은 하부 블록체인에 맞는 형태로 변형될 수 있어야 한다.

따라서 Bifrost는 DApp 을 프라이빗 부분과 퍼블릭 부분으로 나누고(split) 각각을 해당 블록체인 시스템에 해당하는 native 스마트 컨트랙트 언어 또는 중간 언어로 변형(transpilation)하는 접근 방법을 사용한다.

**Abstraction Layers for Universal Programs.** 전통적으로 이기종의 시스템에서 동작하는 프로그램 실행 환경을 만드는 과정에서는 또하나의 virtual machine이나 virtual OS와 같은 추상화 계층(abstraction layer)을 추가하여 구성하는 방법을 많이 수행해 왔다. 하지만, 추상화 계층은 명령어가 간접적으로 실행되면서 전체 성능을 떨어지고 하위 시스템의 기능을 직접적으로 사용할 수 없다는 제약을 가지고 있다.

**Transpilation to Eliminate Abstraction Layers.** Bifrost의 DApp은 universal program들과 비슷한 목적을 가진다. 우리는 DApp이 추상화 계층의 문제를 겪지 않으면서 조합된 블록체인에서 자유롭게 동작하게 하고자 한다. 따라서 Bifrost에서는 "Abstraction without guilt"를 표방하고 있는 multi-stage programming의 접근 방법을 취한다. 고수준 단계에서 작성된 DApp 에서 시작하여 퍼블릭과 프라이빗 부분으로 분리하고, transpilation 과정을 통해서 각각을 목적(target) 블록체인에 맞는 언어로 변형한다. 특히 DApp을 구성하는 스마트 컨트랙트는 일반적인 컴퓨팅 환경에 비하여 코드의 크기가 훨씬 작으며, 기능적으로 완결된(self-containing) 상태를 가지기 때문에, 코드 레벨에서의 변형을 하는 transpilation 과정에 적합하다.

## 3.2 Recipe Language System

Bifrost접근방법은 이중 블록체인에서의 DApp 운영을 가능하게 하지만, 다음과 같은 요구사항을 고려해야 한다.

- 개발자가 DApp을 블록체인에 종속되지 않고 (blockchain-agnostic) 작성할 수 있는 중립적인 방법이 필요하다.
- DApp에서 분리된 두 component는 Linker의 중재에 의해서 동시 실행된다. 하지만 이러한 교차 실행(cross execution)을 지원하기 위해서는 기존의 동기적(synchronous) 명령어의 실행 과정이 비동기적(asynchronous)으로 바뀌는 프로그램 모델의 변화가 필요하다. 따라서 DApp이 이러한 비동기 실행에 대한 변화에 적응할 수 있어야 한다.
- 스마트 컨트랙트의 안전성을 보장하기 위해서는 correctness와 fairness의 문제점을 동시에 고려하며 검증해야한다. 특히 비동기성이 추가된 상황에서 스마트 컨트랙트의 correctness 검증은 더욱 어려운 문제이다. 따라서, 언어 레벨에서의 지원이 필요하다. 스마트 컨트랙트의 의미적 조건 들을 표현할 수 있도록 하여, model checking과 같은 formal method 들의 적용을 도울 수 있어야 한다.

우리는 이러한 요구사항을 바탕으로 근본적으로 DApp 개발자의 프로그램에 대한 접근 방식이나 모델에 대한 변화를 유도하고, 효과적인 변형 과정을 지원하고자 새로운 프로그래밍 언어 체

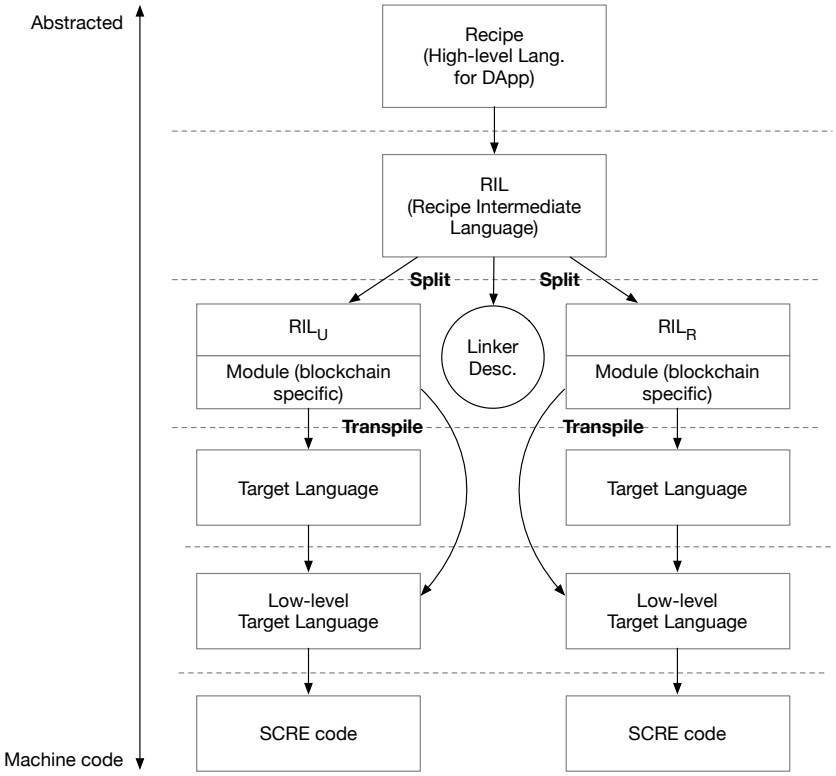


그림 5: Recipe Language System

계를 구성한다. DApp 개발을 위한 고수준 언어인 "Recipe"를 비롯하여 Recipe를 위한 중간언어인 RIL(Recipe Intermediate Language)와 변형과정을 위한 규칙을 함께 정의한다. 그림 5는 Recipe Language System의 구성 및 변화 과정을 보여준다. 그림에서 위에 위치할 수록 abstraction을 많이 포함하고 있고 아래로 내려갈 수록 실제 블록체인 시스템에서 실행되는 실행 코드에 가까워진다.

Recipe Language System은 BIFROST를 위하여 다음과 같은 단계별 언어를 지원한다.

- Recipe: 스마트 컨트랙트를 개발하기 위한 블록체인 중립적인 언어이다. 이중 블록체인 시스템을 고려하여 각 변수와 함수가 수행되어야 할 대상을 지정할 수 있다. 또한 분리된 component 사이의 state의 변화나 함수 호출 과정의 asynchronous 한 특성을 고려한 프로그램 모델을 가지고 있다. 또한 자동화된 보안성 검증을 지원하기 위하여 DApp이 가지는 semantic 정보를 표기할 수 있다.

- RIL (Recipe Intermediate Language): Recipe 언어의 처리 및 분석의 용이성을 위하여 사용하는 중간 언어이다. 자동화된 처리를 위하여 Recipe보다 명확하고 단순화된 문법을 가지고 있다. 내부적으로는 RIL 상태에서 코드의 분리와 연결점 생성등의 변형이 이루어진다. 프라이빗과 퍼블릭 블록체인을 위해 분리된 RIL을 각각 $\text{RIL}_L$ 과 $\text{RIL}_U$ 라고 하며 또한 RIL은 transpile 과정을 위하여 module 시스템을 지원한다.

- Target language: 블록체인 시스템들에서 스마트 컨트랙트를 작성하는데 사용하는 프로그래밍 언어로, 현재 사용되고 있는 Solidity나 Go와 같은 언어를 의미한다. 이것에 대비하여, 이러한 언어들의 build 과정에서 사용되거나 실행 코드에 더 가까운 중간 언어를 low-level target language로 구분하며. LLL이나 LLVM 등이 이에 해당된다.
- SCRE(Smart Contract Runtime Environment) code: 스마트 컨트랙트 실행에 사용되는 코드 상태를 의미한다.

### 3.3 Splitting and Transpilation

Recipe는 언어 레벨에서의 분리와 변형 과정을 지원한다. Recipe 프로그램 코드에서 지정한 정보를 바탕으로 Bifrost는 RIL 코드를 생성하고 분리한다.

**Cross Variables and Functions.** 우리는 양쪽의 component에서 사용되는 변수와 함수를 각각 "cross variable"과 "cross function"으로 정의한다. 일반적으로 퍼블릭 블록체인에서 사용되는 정보들이 cross variable과 cross function으로 설정된다. Cross variable과 function에 대한 정보를 바탕으로 각 component별로 내부에서만 사용되는 local variable 또한 "component local variable"로 구분한다. Cross variable과 cross function은 두 component 사이에서 정의되므로 확정되거나 결과를 얻기 위하여 Linker의 중재가 필요하다. 따라서 cross variable과 cross function에 대한 호출 결과는 *transient* 와 *finalized* 의 두 가지 상태를 가진다. Transient 상태의 경우에는 값의 확정이 이루어지고 있기 때문에, 해당 값을 사용할 수 없다. 해당 값은 finalized 상태로 변경이 이루어 졌을 때에 사용할 수 있게 된다. Bifrost에서는 분리의 과정에서 cross variable로 설정된 변수는 상태를 확인하고 사용할 수 있도록 lock/unlock 기능을 추가한다. 또한, Recipe에서 cross function의 finalized된 호출 처리할 수 있는 callback 방식의 handler 코드를 지원한다.

**Process to Build Dual Components.** 컴포넌트 생성 과정은 블록체인의 특성에 따라 다른 방법이 적용될 수 있지만, 공통적으로는 다음과 같은 절차를 통해서 진행된다.

1. Recipe 프로그램 코드의 표기를 바탕으로 컴포넌트 별로 포함될 변수와 함수의 set을 구분한다.
2. Recipe 프로그램을 RIL 형태로 변형한다.
3. 분석을 통하여 RIL 프로그램 내의 control/data dependency (super-)graph을 생성한다.
4. Private component와 public component에 해당하는 $\text{RIL}_L$ 과 $\text{RIL}_U$ 로 분리한다.
5. 두 component 간의 구분(boundary)를 넘는 control flow와 data flow를 식별(identify)한다.
6. $\text{RIL}_L$ 과 $\text{RIL}_U$ 각각에 대하여 식별된 flow의 연결점(crossing point)에 해당하는 stub code를 추가한다. Cross function은 Linker에 해당 호출이 발생했음을 알리는 stub code로 변경하고 cross variable의 변경은 setter 함수로 변경한다.

7. 연동 과정에서 Linker가 관찰해야 하는 cross function 또는 cross variable 들의 정보 등이 포함되어 있는 연동을 위한 Linker description 정보를 생성한다.

8. 생성된 component의 단계별 결과를 활용하여 DApp 전체 모델에 대한 correctness와 fairness에 대한 안전성을 자동 점검 한다.

9. Blockchain-specific module 시스템과 syntax rule을 적용하여 $\text{RIL}_L$ 과 $\text{RIL}_U$ 를 대상 블록체인의 대표 언어로 변환(transpile)한다.

위의 과정을 통해서 하나의 DApp Recipe에서 최종적으로 블록체인 시스템마다의 SCRE code 들의 모음과 Linker에 전달해야 할 정보인 Linker description을 얻게된다. 이 후 각 SCRE-code component들을 블록체인에 설치하고 Linker description을 Linker에 전달함으로써 DApp 서비스를 운영을 시작한다.

Bifrost은 DApp의 자동분리생성 과정에서 발생할 수 있는 보안의 취약점을 최소화하기 위해서 컴포넌트의 생성과 보안성 점검을 통합하여 수행한다. 또한, 컴포넌트 레벨에서의 보안성과 함께 DApp 레벨에서 컴포넌트들과 Linker의 조합과정에서 발생할 수 있는 문제점을 model checking등의 방식을 통하여 검증한다.

# 4 Linker & BFC System

**Sub-goals of Linkers.** Bifrost는 서로 다른 블록체인을 조합하고, DApp의 분리된 component가 각 블록체인의 장점을 활용하며 하나의 DApp 처럼 동작하도록 한다. §3에서 DApp이 분리되고 변형되어 준비되고 난 이후에 실행과정에서 두 DApp의 연동은 Linker가 담당한다. 즉, Linker는 분리되어 있는 두 개의 component가 정보를 공유하고 통합된 동작을 할 수 있도록 relaying proxy로써의 역할을 한다. 하지만, 블록체인에서 동작하는 DApp은 스마트 컨트랙트 모델을 가지고 있기 때문에 연동의 과정에서 일반 프로그램과 다르게 접근해야 한다. 스마트 컨트랙트는 내부적인 프로그램 state 외에 블록체인을 이용한 permanent storage와 native coin에 대한 balance를 가진다. 따라서 Linker에서는 스마트 컨트랙트 특성과 블록체인이 가지는 특성을 고려하여 아래와 같은 세부 목표를 가진다.

- Transparent Interoperation: Recipe로 작성된 DApp의 두 component가 추가적인 연동에 대한 작업없이 Linker를 통해서 상호운용될 수 있도록 한다.
- Platform-wide, Unified coin (token) System: Bifrost에서 동작하는 DApp을 위해 하나의 coin (또는 token) 체계가 필요하다.
- Relay Availability: DApp의 두 component가 모두 동작하는 동안에는 상호운용을 위한 연결이 항상 유지되어 있어야 한다.
- Adjustable Finality: 블록체인은 consensus 방식과 시스템 특성에 따라서 서로 다른 finality를 가지게 된다. 예를 들어, 작업증명(PoW) 방식의 블록체인 시스템은 확률적인 finality를 가지지

만, 지분증명(PoS) 방식 등을 통하여 즉각적인 finality에 가까운 효과를 얻을 수 있다. 따라서, Bifrost에서 서로 다른 finality를 가지는 블록체인이 조합된 경우에 Linker에서 서로 다른 finality에 대한 절충 기능을 제공해야 한다.

**Interworking on Bifrost.** Linker는 세부 목표를 달성하기 위하여, DApp의 data와 control에 대한 연동 방법과 함께 Bifrost플랫폼 전체에서 사용할 수 있는 새로운 코인인 BFC시스템을 정의한다. 또한, Linker의 논리적/물리적 다중화를 통해서 가용성(availability)를 보장하고, DApp의 분리과정에서 생성되는 Linker description에 finality 옵션을 제공하여 작업의 중요도와 기반 블록체인 특성에 맞는 finality를 적용할 수 있도록 한다.

## 4.1 Interworking via Linkers

Bifrost에서 동작하는 DApp의 공통된 환경을 제공하기 위하여 DApp 실행을 위한 연료 또는 매개체로 사용되는 통합 화폐인 BFC를 정의한다. 그리고 data와 control의 연동을 위하여 Linker가 수행하는 작업을 정의한다.

### 4.1.1 BFC Operation

BFC는 블록체인을 조합하는 Bifrost의 특성을 고려한 token 체계로서, Bifrost생태계 구축의 매개체로 사용된다. (Bifrost생태계는 §6 에서 소개된다.) BFC는 퍼블릭 블록체인의 Token System으로 (e.g., ERC20-compatible token) 운영된다. 내부적으로는 BFC는 Bifrost의 프라이빗 블록체인의 native coin으로 전환되어 분리된 DApp의 실행이나 결제 시스템 구성과 같은 본연의 역할을 수행한다. 퍼블릭 블록체인의 BFC와 프라이빗 블록체인의 native coin으로의 전환은 Linker가 담당하며 전환 비율은 Operator에 의해서 정해진다 (§6.2). 따라서 Bifrost에서 운영되는 BFC는 퍼블릭 블록체인 관점에서는 DApp의 public component들과 Linker들의 거래에서 확인할 수 있다. Bifrost Container 내의 프라이빗 블록체인 관점에서는 BFC형태가 아닌 native coin의 형태로 보이게 된다.

BFC는 Bifrost의 DApp이라면 누구나 사용할 수 있지만, DApp도 스스로 퍼블릭 블록체인에 자신만의 token 또는 coin 시스템을 가질 수 있다. 이러한 DApp들은 자신의 public component를 통해서 자신뿐만 아니라 다른 token 시스템의 기능도 활용할 수 있다.

### 4.1.2 States

Bifrost에서 Linker가 고려하는 state 들은 다음과 같다.

- Cross Variables: Permanent storage에 저장되는 방식은 블록체인 시스템의 모델과 구현방식에 따라 다르지만, DApp을 개발하는 고수준 언어에서는 변수로 추상화되어 나타난다. 따라서 Linker에서는 cross component 사이에서 유지해야하는 storage를 변수로 통일하여 취급한다.
- Private Native Coin Balance: (프라이빗 블록체인에서 사용되는 coin을 private native coin으로 정의할 때) private coin에 대한 balance를 의미한다.

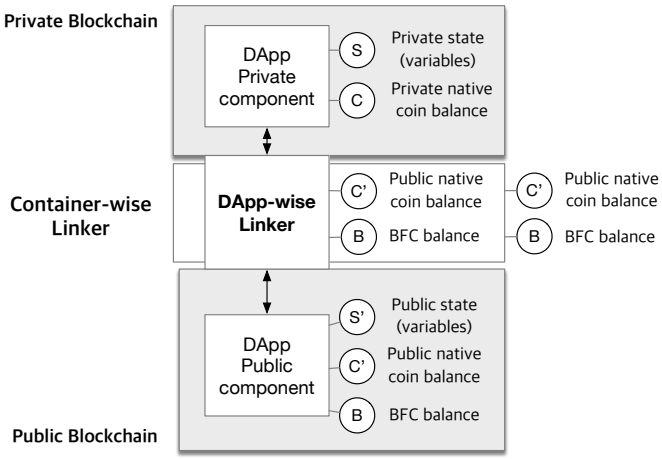


그림 6: DApp과 Linker의 내부 state 구성

- Public Native Coin Balance: 위와 유사하게 퍼블릭 블록체인에서 사용되는 native coin의 balance를 의미한다.
- Component Addresses: DApp component가 위치한 블록체인 시스템에서의 address 정보를 관리한다.
- BFC Balance: Bifrost에서 사용되는 BFC의 주소별 balance 정보이다. BFC는 퍼블릭 블록체인에서 동작하기 때문에, 퍼블릭 블록체인에서 존재하는 주소에 대한 BFC의 balance를 의미한다.

#### 4.1.3 Actions

Linker를 통해서 연동을 해야 하는 action들은 다음과 같은 작업들을 포함한다.

- Cross-function Call: 하나의 DApp에 속하지만 서로 다른 component에 위치한 두 함수가 서로를 호출하는 경우
- Cross-status Update: 하나의 DApp에 속해 있는 한쪽의 component에서 두 component가 서로 공유하는 상태 정보(cross variable)을 변경하여 다른 component에도 update가 필요한 경우

### 4.2 DApp-wise and Container-wise Linkers

Linker는 기능적으로 블록체인의 연결을 담당하지만, 수행하는 역할에 따라서 그림 3과 4에서와 같이 두 가지 종류로 구성된다.

- DApp-wise Linker: DApp 별로 생성되어서 해당 DApp의 연결을 담당한다. DApp의 재구성 과정에서 분리된 component들간의 연동 정보인 Linker description을 바탕으로 연동을 지원한다.

- Container-wise Linker: 하나의 Container에서 동작하는 여러 DApp-wise Linker 들을 총괄하는 Linker로서, DApp-wise Linker 들을 생성하고 관리한다. 또한, DApp-wise Linker 들이 DApp public component로의 transaction을 발생시킬때 사용하는 public native coin의 balance 재충전도 담당한다.

그림 6에는 Linker의 종류별로 저장해야 하는 state 정보들이 나타나있다. DApp private component는 내부적으로 사용하는 variable 들과 private native coin balance를 유지한다. DApp public component는 퍼블릭 블록체인에 저장되어야 할 정보를 담고 있는 variable 들과 public native coin의 balance를 유지한다. 또한, DApp이 보유하고 있는 BFC의 balance도 이 DApp의 public component의 주소에 할당(assign)된다. Linker의 경우에는 DApp의 정보를 위한 variable을 가지지 않지만, public native coin과 BFC를 위한 balance를 가진다. DApp-wise Linker의 BFC balance는 BFC를 private component의 private native coin balance로 변환하기 위하여 사용한다. Public native coin balance는 private component에서 public component로의 cross function call을 발생시키는 경우에 퍼블릭 블록체인에서 transaction을 생성하기 위한 비용으로 사용한다. Container-wise Linker도 유사하게 두 종류의 balance를 유지한다. Container-wise Linker의 경우에는 직접적으로 transaction을 호출하기 보다는 Operator의 입장에서 DApp-wise Linker의 public coin balance를 일정수준으로 유지시켜주는 용도로 사용된다.

## 4.3 Relaying State of Cross Variables

Linker는 state에 대한 연동을 위해서 cross variable의 변경된 내용이 양쪽의 component에 올바르게 반영될 수 있도록 한다. 두 component에서 하나의 cross variable에 대해 동일한(consistent) 정보를 유지하기 위하여 두 변수의 내용을 동기화하거나 변수의 접근을 함수화 하는 두가지 접근 방법을 사용한다.

첫번째 방법은 cross variable을 양쪽에 위치시키고, DApp의 분리 및 재구성 과정에서 해당 변수의 값을 정의(definition)하는 할당문(assignment statement)을 setter 함수로 대치한다. Setter 함수는 해당 변수가 transient 상태에 있음을 설정하고 Linker에게 값의 변경을 알린다. Linker는 이러한 state 변화가 반영되었음을 확인하고 양쪽 component에 알린다. 두번째 방법은, cross variable의 사용 빈도가 적은 경우에는 Recipe의 생성(build)단계에서 해당 변수에 대한 사용을 getter/setter 방식으로 변경한다. 이러한 경우에는 cross variable을 한쪽 component의 local variable 처럼 사용할 수 있고, Linker의 state 동기화를 cross function call에 대한 문제로 단일화 할 수 있다. 따라서 cross variable이 한쪽 component에서만 주로 사용되는 경우에 유리하다.

## 4.4 Linker Availability

연합(federation)의 형태로 같은 프라이빗 블록체인 네트워크를 구성하고 있는 Container들은 분리되어 있지만, 동일한 기능을 하는 Linker들의 조합으로 구성되어 있다. 즉, Container 마다 동일하게 동작하는 DApp-wise Linker들이 있는 셈이다. 한 DApp에 대한 DApp-wise Linker는 여러 Container

에서 복제되어 있는 경우에도 블록체인 내부에서는 하나의 계정을 통해서 동일한 동작을 수행하게 된다. 따라서, 블록체인 시스템 내에서는 단일한 주체에 의한 복제된 동작으로 표현된다. 이러한 여분의 동시 동작을 통하여 Linker 기능의 가용성을 보장한다.

## 4.5 Relaying Cross-function Calls

Linker는 한쪽의 component에서 cross function에 대한 요청(call)을 반대편의 component의 해당 함수에 전달한다. Recipe의 분리과정에서, Bifrost는 cross-function call 들을 Linker에 알림을 보내는 stub code로 변경한다. 이더리움과 같이 블록체인 시스템에서 이벤트 시스템을 가지고 있는 경우에는 Linker가 해당 이벤트들에 대한 구독(subscription) 형태로 이러한 요청을 확인할 수 있지만, 이벤트 시스템이 없는 경우에는 component 주소에 대한 관찰을 통하여 함수 호출의 발생을 확인한다.

그림 7은 DApp-wise Linker를 통해서 양쪽의 component 사이의 state 및 action의 전달 과정의 예를 보여준다. Intra-DApp call의 경우에는 DApp의 재구성 과정에서 해당 함수에 대한 호출과정을 Linker의 전달 과정으로 코드상에서의 대치가 가능하다. 하지만 외부 DApp에서 다른 component에 있는 함수를 호출하는 inter-DApp call의 경우에는 해당 함수와 동일한 signature를 가지는 함수를 구성하고, 호출되는 경우 바로 Linker에 전달한다.

Bifrost의 DApp 들은 다중 component로 구성되기 때문에 모든 실행과정이 비동기적으로 (asynchronously) 일어난다는 것을 가정해야한다. 따라서 Recipe 프로그램 모델에서는 cross function call에 대하여 finalized 결과의 실제 반영을 처리하는 callback 형태의 handler를 추가하도록 한다.

## 4.6 Relaying Coins

BFC가 platform-wide coin으로써 Bifrost가 동작하는 어디에서든 사용될 수 있지만, 실제 DApp의 실행과정에서는 component에 맞게 변환되어야 한다. 한 블록체인의 asset을 외부나 다른 시스템에 옮겨서 사용하기 위해서, 기존에는 한쪽의 asset을 burn 하거나 lock을 거는 방식으로 고정시킨 후 다른 쪽 블록체인에서 asset을 사용할 수 있게 하는 atomic swap 형태의 기법들이 사용되었다. Bifrost의 Linker도 비슷한 방식으로 coin을 이동하지만, 상황은 더 단순하다. Container operator에 의해서 관리되는 DApp-wise Linker의 주소로 전송한 BFC만이 프라이빗 블록체인의 native coin으로 직접 변환되어, 미리 지정된 private component의 주소로 전송되는 형태이다. 따라서 Linker와의 hashed timelock contract 방식을 통해 BFC를 프라이빗 블록체인에 안전하게 전달할 수 있다.

Linker를 통한 coin의 이동을 고려하여, DApp의 public component와 Linker는 분리된 balance를 가진다. DApp 자체에서 소유하고 있는 퍼블릭 블록체인의 balance는 public component에서 관리하고 Linker의 balance는 coin의 환전이나 transaction의 수수료(또는 gas) 지급의 한정적인 용도로만 사용한다.

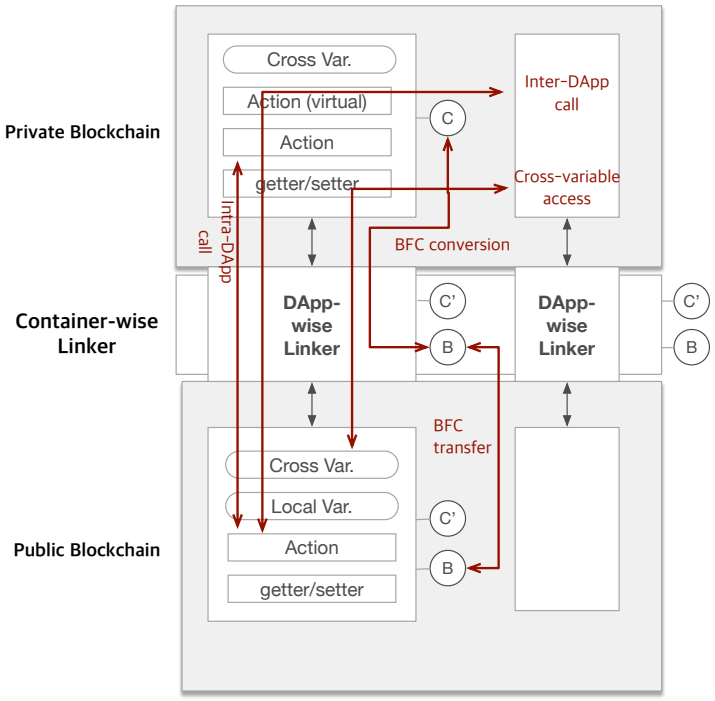


그림 7: DApp-wise Linker를 통한 state와 action의 연결의 예

### 4.7 Linker Description

Linker description은 DApp 프로그램이 분리되는 과정에서 자동으로 생성된다. Linker description에는 DApp-wise Linker에서 관리/관찰해야 하는 cross variable과 cross function들에 대한 정의와 signature가 들어간다. 또한 cross variable과 function의 상태라 finalized 되고 난 이후에 실행해야할 함수들의 목록도 함께 포함된다. 특히, 각 결과가 실제 finalized 되었다는 것을 결정하기 위하여 어떠한 조건이 필요한지를 명시할 수 있다. 예를 들어 작업증명을 사용하는 블록체인의 경우에는 결과가 finalized 되었다고 component에게 알려주기 전에 필요한 confirmation을 명시한다.

## 5 Container

### 5.1 Multiple Containers for Service Availability

Bifrost에서 퍼블릭 블록체인과 함께 프라이빗 블록체인 환경을 구성하기 위해서는 다음과 같은 점을 고려해야 한다.

- 프라이빗 블록체인 시스템의 기초적인 가용성과 안전성을 확보하기 위하여 다중의 노드 구성이 필요하다.

- 프라이빗 블록체인 시스템을 이용하여 복수 기관이 참여하는 연합 블록체인을 구성하는 경우에는, 연합 네트워크의 요구사항을 지원하기 위해 다중 노드를 유연하게 설정할 수 있어야 한다.
- Linker와 같이 Bifrost에서 제공하는 서비스 또한 가용성의 확보가 중요하다.

우리는 프라이빗 블록체인 노드와 Bifrost의 기능 모듈이 모두 다중화를 통해 가용성을 보장해야 한다는 점에 착안하여 Container 구조를 구축하였다. Bifrost는 프라이빗 블록체인 노드와 Linker와 같은 DApp 연동 및 이용을 위한 기능들을 묶어 하나의 Container 로 구성한다. 그리고 서비스 가용성을 위하여 하나의 프라이빗 블록체인 네트워크에 포함된 Bifrost Container 들이 동일한 작업을 동시에 수행한다. 또한 Container manager를 통하여 가용성 확보와 효율성의 trade off를 고려하며 container의 동적 관리를 할 수 있도록 지원한다.

## 5.2 Container Manager

그림 3에서 나와있는 것처럼 하나의 Container에는 하나의 프라이빗 블록체인 시스템을 담는 것을 기본으로 하고 있다. 전체 플랫폼에서 서비스의 안전성을 높이거나 독립된 공간에 별도의 프라이빗 블록체인을 구성해야 하는 경우에는 Container manager를 통하여 Container를 추가적으로 생성하여 사용할 수 있다. Container manager는 각 Container에 대한 독립성과 서로간의 연결성을 보장한 연합 블록체인 구성을 지원한다. 또한 standalone 방식의 Container manager를 통해서 완전히 독립된 Container를 구성할 수도 있다.

**Creation by Replication.** 하나의 연합에 속하는 새로운 container가 추가되는 경우에는, 블록체인의 특성상 프라이빗 블록체인의 state 정보는 자동적으로 복제(replicate)된다. 하지만 Bifrost에서 지원하는 Linker의 availability를 보장하기 위하여 Linker의 해당하는 기능도 복제한다. 즉 한 Container가 가지고 있는 Linker description 등의 정보와 Linker에서 사용하는 account의 접근 정보가 복제된다.

## 5.3 Operator

Operator는 서비스 제공자로서 해당 Container를 DApp 서비스 운영자에게 제공하는 주체를 의미한다. Bifrost는 Container 마다의 자율성을 제공하기 때문에 third-party의 Operator가 단독(standalone) 또는 공유 Container들을 운영할 수 있다. Operator에 따라서 서로 다른 프라이빗 블록체인을 가지는 Container가 생성될 수 있고, 기반 퍼블릭 블록체인과의 조합에 따라서 다양한 서비스 조합이 가능해진다. 또한, Operator는 Container의 특성과 목적에 따라서 자신의 서비스 과금 정책을 설정할 수 있다. (Operator의 자세한 역할은 §6.2에서 설명한다.)

# 6 Bifrost Ecosystem

## 6.1 Bifrost Ecosystem Overview

Bifrost의 새로운 DApp 플랫폼을 실제 활성화하기 위해서는 참여자들의 협력에 의해서 유지되는 자생적 생태계가 필요하다. Bifrost는 잘 정의된 incentive 방법을 통한 참여자들간의 decentralized governance 구조를 통해서 지속가능한 생태계를 만들고자 한다.

Bifrost의 주된 참여자는 다음과 같이 구분할 수 있다.

- Bifrost Operator: 선택한 퍼블릭 블록체인 위에서 하나 또는 복수의 Container로 구성된 프라이빗 (또는 연합) 블록체인을 운영하며 DApp Provider에게 서비스를 제공한다.
- DApp Provider: Bifrost Operator 를 선택하여 자신의 DApp을 운영한다.

우리는 Bifrost를 통해 다음과 같은 생태계를 만들고자 한다.

1. Bifrost Operator는 DApp Provider에게 합리적인 운영의 비용을 요청하며, DApp provider는 그 비용을 지불한다.
2. Bifrost Operator는 DApp Provider에게 합의된 서비스를 공정하게 제공한다.
3. Bifrost시스템은 DApp Provider가 Operator를 선택한 이후에도 자유롭게 최적의 Container로 이동할 수 있도록 지원하여, 경쟁을 통해 Operator들의 지속적인 발전을 유도한다.
4. Bifrost생태계가 활성화될 수록 참여자들이 경제성과 시스템 안전성 등의 측면에서 이득을 볼 수 있도록 한다.

이러한 생태계를 운영하기 위한 매개체로서 Bifrost에서 BFC token 을 구성한다. BFC는 다양한 블록체인의 조합이 사용되는 Bifrost의 특징을 고려하여 platform-wide 하게 사용될 수 있는 토큰이다. DApp Provider와 Operator 사이의 거래나 참여자들의 incentive 체계 구성을 위한 도구로서 사용된다.

## 6.2 Mechanisms

Bifrost Operator는 서비스를 제공하면서 DApp Provider에게 사용료를 받아 이익을 얻지만, 동시에 정당한 비용을 청구하고 시스템을 충실하게 운영해야하는 책임을 가지게 된다. 반대로 DApp Provider는 자신에게 합리적인 가격에 가장 양질의 서비스를 제공해주는 Operator를 선택하기 위해 노력한다. Bifrost의 생태계에서는 이러한 서로의 목적을 달성하기 위하여 BFC를 통해서 정보의 비대칭성(information asymmetry)이 없는 경쟁시장 구조를 이용한다.

**Main Process.** Operator들은 가격과 특징을 내세운 서비스 조건을 홍보하고, DApp Provider는 세심한 review 과정을 거쳐서 자신에게 가장 적합한 Operator를 결정한다. DApp의 선택 결과는 사용료 지불을 위해 각 Operator에게 staking하고 있는 BFC의 양으로 표현된다. 또한 Bifrost는 손쉬운 Operator간의 migration을 지원하기 때문에, DApp Provider는 해당 Operator가 제공하는 서비스의 보안적이나 성능적 결함이 있는 경우 언제든지 다른 Operator를 선택하여 이동할 수 있다. 따라서, DApp Provider의 BFC staking 정보는 각 Operator에 대한 지지도 또는 신뢰도의 척도로 작용할 수 있다. 우리는 이 정보를 "Operator credit"이라 부른다. BFC는 퍼블릭 블록체인 위에서 동작하므로, Operator credit 현황에 대한 검증된 정보를 투명하게 확인할 수 있다. Bifrost는 이러한 정보들을 손쉽게 확인할 수 있는 툴(tool)들을 함께 제공하여 정보의 대칭성을 보장한다.

**Service Plan Publication.** Bifrost Operator는 자신의 service plan을 공표한다. Service plan에는 다음의 내용이 포함된다.

- Operator의 기본 정보: 해당 Operator가 어떠한 블록체인을 조합한 서비스를 운영하고 있고 어떠한 Container들을 운영하고 있는지에 대한 시스템 정보를 제시한다. (추가적으로 지원하는 security mechanism 이나 성능 향상을 위한 지원 시스템들도 표기할 수 있다.)
- Pricing Function: DApp의 사용 비용을 함수로 표현한다. Operator에 따라서 DApp의 사용 정도(transaction의 복잡도나 public component의 사용 빈도)에 따라 BFC를 지불할 수도 있고, 특정한 양의 BFC를 staking하게 할 수도 있다. 따라서 Pricing function의 정의를 위하여 DApp의 transaction의 복잡도와 개수, 서비스 이용을 위한 DApp의 BFC staking 양 등을 파라미터화하여 제공하고, Operator는 주어진 파라미터를 사용하는 pricing function을 제시한다.
- BFC Staking per DApp: Operator는 악의적인 운영을 방지하기 위하여 동작하는 DApp의 숫자에 비례하는 양의 BFC를 staking 해야 한다. 이는 일종의 security deposit의 역할로서, 해당 Operator에게서 약속된 서비스를 받지 못하는 경우 blame 과정을 통하여 이 금액이 DApp Provider에게 지불된다. DApp Provider는 현재 Operator가 staking 하고 있는 BFC의 양과 이 정보를 바탕으로 몇 개의 DApp이 동작하고 있으며, 가능한 자리(slot)가 얼마나 남았는지를 확인 할 수 있다.

**Optimal Operator selection.** Bifrost내에 있는 모든 Operator의 현황은 service plan과 함께 제공된다. DApp Provider가 Operator를 선택하면 service plan에서 요구하는 양을 우선 선불의 형태로 staking 하고, pricing function을 통해서 비용이 실제 Operator에게 지불된다. DApp Provider들의 선택은 해당 Operator에 얼마나 많은 양의 BFC가 모여있는지로 판별할 수 있다(Operator credit). 즉, Operator credit이 각 Operator의 신뢰도를 보여주는 voucher의 역할을 한다. 따라서, Operator는 양질의 서비스의 대가로 DApp에게 더 많은 BFC staking 양을 요구함으로써, 선택에서 유리한 입지를 차지할 수 있다.

**Penalty and Blame** Operator는 DApp Provider의 항의 과정을 통하여 충실하지 못한 운영에 대하여 처벌받을 수 있다. 기본적으로는 mechanical blame 방법을 지원하고자 한다. DApp Provider는 Operator를 선택하면서 초기 service plan에 따라 지켜야할 성능의 파라미터를 service level agreement 정의한다. Operator가 이를 위반하는 경우 한 DApp에 해당하는 staking 양의 BFC를 smart contract등을 이용한 자동적인 절차를 통해서 DApp Provider에게 지불되도록 한다. 이러한 방식의 보완책으로서 Operator committee의 투표에 의한 방식을 고려할 수 있다. 모든 Operator들은 자동적으로 Operator committee의 회원이 되고 자신의 Operator credit 만큼의 투표권을 가지게 된다. DApp Provider가 일정 BFC를 소비하여 Operator committee에 특정 Operator에 대한 blame report를 제시하면 정해진 기간내에 Operator committee들이 수락/거절의 투표를 한다. Blame report가 수락된 경우에는 앞의 경우와 마찬가지로 한 DApp에 해당하는 staking 양의 BFC를 지불한다.

## 6.3 Bootstrapping

Bifrost의 생태계는 참여자가 늘어날 수록 전체적인 플랫폼의 안전성과 효율이 높아지는 네트워크 효과를 가지고 있다. Container의 숫자와 제공되는 서비스의 질이 비례하는 효과를 고려한다면 네트워크 효과뿐 아니라 지원되는 메인넷 프로토콜의 종류나 다양성 측면에서도 생태계 효과는 뚜렷하게 나타난다. 하지만, DApp Provider와 Bifrost Operator 서로가 상대방이 번성할 수록 자신도 이익을 보는 "two-sided market"에 해당되기 때문에, 네트워크 효과를 점화(trigger)하기 위한 초기의 노력이 필요하다. 따라서 Bifrost에서는 다음과 같이 단계적인 접근을 하고자 한다.

- 초기 단계에서는 Bifrost에서 주축에 되어 DApp의 원활한 작동을 지원하는 Container 운영을 주도한다: Container운영의 기본이 될 수 있는 Pricing function 지원 및 초기 모델 지원한다.
- Bifrost와의 파트너 관계에 있는 기관들이 Operator로서 Bifrost의 Container들을 운영한다.
- 검증된 third-party Operator가 지원하는 경우 초기 운영을 위한 BFC를 지원하여 참여를 독려한다.
- Bifrost생태계가 성숙되어 가면 Operator의 진입 장벽을 낮춰 더 나은 Container 서비스 운영의 경쟁 시장 체제를 유도한다.

Bifrost의 토큰, BFC는 초기 일정량 발행되며, 이후의 채굴은 존재하지 않는다. Bifrost의 목적대로 DApp들이 유치되어 그 생태계가 조성되면 초기단계의 Staking에 필요한 BFC이외에 Container Operator에게 지급되는 Fee형태의 BFC에 대한 수요도 증가하게 된다. Bifrost의 네트워크 효과는 전체적인 플랫폼의 안정성과 나은 서비스의 제공을 보장하며 또한 BFC의 가치를 보장한다.

# 7 Conclusion

블록체인의 서비스에 대해 관심있는 사람들의 흔한 첫번째 질문은 현재까지 나온 가장 대중적인 “Killer App”이 무엇이냐는 것이다. 이 질문에 대해 누구나 공감할 수 있는 대답을 할 수 있는 순간이 블록체인 기술이 대중화되는 시기일 것이다. 다시 말해, 블록체인이 세상을 변화시킬 수 있는 첫 걸음은 대중이 그 서비스를 체험할 수 있는 DApp의 보편화인 것이다.

Bifrost의 핵심은 현재 DApp이 안고 있는 문제를 해결해주고 그 생태계의 현실적인 활성화를 지원하기 위해 속도와 확장성에 대한 해결책을 제시하는 것이다. 단, 우리는 수많은 프로토콜의 길을 가지 않고 현재의 시스템을 이용한다. 현재 개발되고 운영되고 있는 퍼블릭 블록체인과 프라이빗 블록체인 시스템들은 저마다의 장점을 가지고 있다. 우리는 Bifrost를 통해서 실제 DApp의 서비스에 이러한 장점들을 동시에 제공하고자 한다. Bifrost의 시스템은 새로운 블록체인의 등장에 맞추어 계속 확장될 예정이다. 이를 통해서 Bifrost의 DApp 개발자들은 기존 서비스의 큰 변화없이 새로운 블록체인으로 이동하거나 필요한 기능만을 쉽게 적용할 수 있다.

Bifrost가 혁신적인 이유는 DApp의 개발환경에 가장 큰 문제인 플랫폼 리스크를 근본적으로 해결할 수 있는 가장 효과적인 솔루션을 제공하기 때문이다. 이는 DApp의 한계를 넓히는 결정적인 계기를 제공할 것이며, 궁극적으로는 블록체인 기술이 대중에 연착륙되는 계기를 만들 것이다.