산업융합연구 제18권 제6호 pp. 147-154, 2020
ISSN 2635-8875 / e-ISSN 2672-0124
DOI : https://doi.org/10.22678/JIC.2020.18.6.147

효율적인 블록체인 교육을 위한 실습프레임워크 설계

김도규
성결대학교 정보통신공학과 교수

Design of Lab Framework for Effective Blockchain Education

Do-Kyu Kim
Professor, Dept. of Information and Communication Engineering, SungKyul University

요 약 상이한 특성을 가진 퍼블릭 블록체인과 프라이빗 블록체인에 대한 전체적인 동작을 통합적으로 교육하는 것은 용이하지 않다. 현재 블록체인의 개념과 동작을 위한 교육은 대부분 비트코인, 이더리움과 같은 퍼블릭 블록체인을 대상으로 하고 있다. 그러나 실제 기업환경에서는 사용자에 대한 인증을 통하여 기업 데이터에 접근통제를 하기 때문에 하이퍼레저패블릭(HLF)과 같은 프라이빗 블록체인이 활용되고 있다. HLF 기반으로 교육하는 경우에는 피어, 오더러, 채널 등 퍼블릭 블록체인에 없는 다양한 구성요소에 대한 이해가 필요하다. 본 논문에서는 이와 같은 구성요소의 기능과 동작에 대한 효율적이고 체계적인 이해를 위하여 실습프레임워크를 설계하였다. 설계된 프레임워크은 HLF 네트워크제어 체인코드제어 탈중앙화소프트웨어(Decentralized Application, dApp) 제어기능으로 구성되어 있다. 프레임워크을 통하여 네트워크을 구성하고 체인코드의 배포 및 활성화 후 dApp 실행과정을 단계적으로 확인하였고 블록체인 서비스를 위한 전체 흐름을 이해하는 것이 매우 용이하였다. 또한 향후 네트워크의 확장 등에도 전체 흐름에 대한 체계적인 이해가 가능할 것으로 기대한다.

키워드 : 하이퍼레저패블릭, 체인코드, 탈중앙화 어플리케이션, 도커 컨테이너, 프레임워크

Abstract It is difficult to educate the overall operation of public and private blockchains with different characteristics. Recently, most education for blockchain is targeted at public blockchains such as Bitcoin and Ethereum. However, in an actual business environment, a private blockchain such as HyperLedger Fabric is used because access to corporate data is controlled through user authentication. In the case of HLF-based education, it is necessary to understand various components that are not in the public blockchain, such as peers, orderers, and channels. In this paper, a lab framework for HLF is designed for an efficient and systematic understanding of the functions and operations. The framework consists of HLF network, chaincode, and decentralized software control functions. Through the framework, the network configuration, distribution and activation of chaincode, and dApp execution process were checked step by step, and it was very easy to understand the overall flow for blockchain services. In addition, it is expected that a systematic understanding of the overall flow will be possible even in future network expansion.

Key Words : Hyperledger Fabric, Chaincode, Decentralized Application, Docker Container, Framework

1. 서론

4차 산업혁명의 근간이 되는 신뢰 기반의 서비스 제공이 가능한 블록체인 기술은 2008년 사토시 나카모토의 “Bitcoin: A Peer-to-Peer Electronic Cash System"이라는 논문을 통하여 등장하였다[1]. 블록체인 기술이 처음 적용된 비트코인 네트워크는 P2P 환경에서 TTP(Trusted Third Party) 없이 노드들 간의 거래가 가능하도록 설계된 시스템이다. 이것은 노드 간의 거래 정보들을 모아 블록을 구성하고, 모든 노드의 합의과정 (Consensus Algorithm)을 통하여 동일한 블록을 공유하는 방식이다. 이때 합의를 위한 방식은 PoW(Proof of Work), PoS(Proof of Stake), DPoS(Delegated PoS)와 HLF(HyperLedger Fabric)

*Corresponding Author : Do-Kyu Kim(dkkim@sungkyul.edu)
Received December 3, 2020 Revised December 18, 2020
Accepted December 20, 2020 Published December 31, 2020

에서 사용하는 Ordering 서비스 (v1.4은 Kafka, v.1.4.1 이상은 Raft Ordering Service) 등과 같은 다양한 방식이 존재한다[2].

블록체인 네트워크에 누구나 참여할 수 있는 경우와 그렇지 않은 경우에는 퍼블릭과 프라이빗 블록체인으로 구분할 수 있고, 그리고 시스템에 접근할 수 있는 범위에 따라 허가형과 비허가형 블록체인으로 구분할 수 있다(Table 1). 이러한 블록체인의 성격에 따라 TPS(Transaction per Second)로 측정되는 성능, 블록을 생성하기 위한 절차 등에 있어서 많은 차이가 발생하게 된다.

Table 1. Public Blockchain vs Private Blockchain

Characteristic	Public blockchain	Private blockchain
Access level	Anyone	Organization
Participation	Permissionless Anonymous	Permissioned Known identities
Consensus	PoW, PoS, ...	Pre-approved participants
Performance	Low TPS	High TPS
Operation	Order, Execute	Execute,Order,Validate
Platform	bitcoin, Ethereum	HyperLedger Fabric

이러한 상이한 특성을 가진 2가지 타입의 블록체인에 대한 전체적인 동작을 통합적으로 교육하는 것은 용이하지 않다. 현재 블록체인의 개념과 동작을 위한 교육은 대부분 비트코인, 이더리움 기반으로 이루어진다. 이것은 블록체인의 개념이 비트코인에서 시작되었고 스마트컨트랙트(SC)의 구현은 이더리움 환경에서 시작되었기 때문이다. 퍼블릭 블록체인 기반의 실습교육에서 필요한 구성요소는 테스트 네트워크, SC의 작성, 컴파일, 배포, dApp(Decentralized Application) 테스트를 위한 환경이 필요하다(Table 2).

Table 2. Public Blockchain Test Environment

Components	Test Environment
Blockchain Network	Ethereum mainet, Ropsten, Geth, Ganache, Javascript VM in Remix
Smar Contract	Remix, Truffle suite
Wallet	Metamask
dApp Developemnt	Truffle suite, Nodejs, React, EJS, Express ...

그러나 HLF과 같은 사설블록체인을 실습하는 경우에는 Peer (Endorser, Anchor, Committing Peer), Orderer, MSP(Member Service Provider), Channel 등 퍼블릭 블록체인에 없는 여러 구성요소에 대한 이해가 필요하다[3]. HLF의 네트워크 구성은 기업 환경에 적합한 구조이고 따라서 각 구성요소가 클라우드 환경에서 컨테이너 기반으로 동작하도록 환경을 구성하는 것이 효율적이므로 도커 컨테이너 환경에 대한 이해도 매우 중요하다.

본 논문에서는 HLF 기반에서 효율적인 블록체인 교육을 위한 실습을 단계별로 진행할 수 있도록 실습 프레임워크를 설계하였다. 프레임워크를 통한 단계별 진행과정을 확인하면 체인코드가 실행되는 흐름을 용이하게 이해할 수 있고, 향후 블록체인 네트워크가 확장되는 경우에도 체계적인 이해가 가능할 것으로 기대한다. 2.1절은 퍼블릭 블록체인의 특성을 기술하고, 2.2절은 실습교육 프레임워크의 동작 환경인 HLF 블록체인의 동작에 대하여 기술한다. 3장에서는 블록체인 교육을 효율적으로 수행하기 위한 실습프레임워크의 구조에 대하여 기술하고, 4장에서는 실습 프레임워크의 구현과 동작결과를 기술하였다. 5장에서는 결론 및 향후 연구를 방향에 대하여 기술하였다.

2. 블록체인 기술

2.1 퍼블릭 블록체인의 동작과 특징

퍼블릭 블록체인은 크게 비트코인과 이더리움 블록체인으로 대표된다. 비트코인은 암호화폐의 거래가 핵심 기능인 반면에 이더리움은 이더의 거래뿐만 아니라 SC를 실행할 수 있는 최초의 블록체인 플랫폼으로서 ERC-20, ERC-721 토큰을 생성하여 거래할 수 있다[4,5].

트랜잭션의 처리과정은 다음과 같다.

① 비밀키를 보관하고 있는 전자지갑을 통하여 트랜잭션 발생

② 생성된 트랜잭션을 인접한 노드에 전파

③ 노드에서 트랜잭션의 내용을 검증한 후 트랜잭션풀(Pool)에 추가

④ 채굴노드는 트랜잭션풀에서 수수료가 높은 트랜잭션을 추출하여 블록을 구성

⑤ 채굴노드는 주어진 조건을 만족하는 블록 해쉬 값을 찾기 위해 머클루트 해쉬 값, 이전 블록의 해쉬 값, 넌스 값을 조합하여 해쉬 값을 구한다. 계산된 해쉬 값이 주어진 난이도를 만족하지 않으면 넌스 값을 변경하면

서 반복하여 해쉬 값을 계산한다. 넌스 값을 찾으면 채굴노드의 코인베이스 주소로 암호 화폐로 보상이 주어진다. 이것을 작업증명(PoW)라고 한다(Fig. 1).

⑥ 블록을 전파하고 각 노드는 블록 검증을 수행

⑦ 가장 긴 체인에 블록 추가

⑧ 채굴된 시간에 따라 적절하게 난이도를 조정하고 ① 번 과정으로 돌아가서 새로운 트랜잭션을 생성한다.

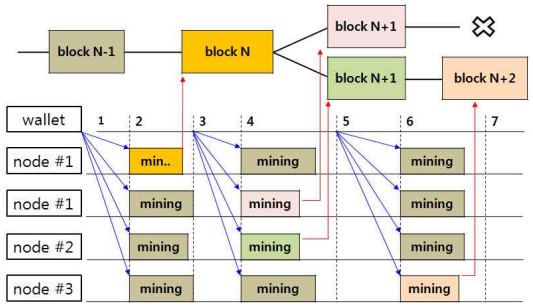

Fig. 1. Mining Block

Table 3에 비트코인과 이더리움의 공통점과 상이한 점을 보여주고 있다. 특히 이더리움 플랫폼은 트랜잭션에 의해 상태가 변하는 상태머신 개념을 도입하였다[6]. 이더리움 클라이언트가 설치된 모든 노드에는 SC를 수행하는 EVM이 동작하고 실행의 결과로 상태가 변경된다. 또한 생성한 트랜잭션은 이더 송금을 하거나 SC을 배포하고 실행하도록 하여 상태를 변경하게 된다. 또한 이더리움에서는 이더를 기반으로 ERC-20, ERC-721 규격에 따라 작성된 스마트컨트랙트 기반의 토큰 발행을 통하여 부가적인 서비스를 제공할 수 있다[7].

Table 3. Bitcoin vs Ethereum blockchain

Characteristic	Bitcoin	Ethereum blockchain
Consensus	PoW	PoW, PoS
Chain rule	Longest chain	Heavist chain
Mining	ASIC-based	GPU-based
SC	No	Yes, Run in EVM
Account	Bitcoin address	EOA, CA
Transaction	BTC transfer	Ether transfer invoke SC
Balance	$\sum_i (UTXO)_i$	Balance in Account
Block generation	≈ 1 blk/10min	≈ 4 blks/min
Token	No	ERC-20, ERC-721

2.2 하이퍼레저패블릭의 구성요소 및 동작

HLF은 기관 혹은 컨소시엄 기반으로 동작하도록 설계된 플랫폼으로서 MSP, Peer (Endorser, Anchor, Committing, General), Orderer, Channel 등으로 구성되어 있다[8,9]. 각 구성 요소가 수행하는 기능은 다음과 같다(Fig. 2).

① MSP : 사용자의 신원 인증 기능을 제공한다. 이를 위하여 인증서의 발급 및 유효성 검사를 수행한다. 개발 시점에는 Fabric CA를 사용하고, 실제 운용에서는 외부 CA를 사용한다. Global MSP와 Local MSP 2 종류로 나누어진다. Channel MSP와 Network MSP는 Global MSP로, 블록체인 네트워크에 참여한 모든 구성원에 적용된다. Global MSP에 정의된 내용에 따라서 블록체인 네트워크가 구성된다.

- Network MSP : 네트워크에 참여하는 조직(Org)의 MSP를 식별한다. 어떤 조직이 블록체인 네트워크에 참여할지 정하고, 네트워크 관리권한을 가진 참여자를 식별한다.
- Channel MSP : 특정 채널에 참여하는 조직(Org)의 MSP를 식별하며, 채널의 관리 권한을 Channel Policy 등으로 정의한다.

② Peer : Endorser, Anchor, Committing

- Endorsing Peer : 트랜잭션의 유효성을 검증하고 Read/Write Set을 반환하는 기능을 수행한다.
- Anchor Peer : Orderer가 전파는 트랜잭션을 다른 피어에 전파하는 기능을 수행한다.
- Committing Peer : 트랜잭션에 의해 CouchDB에 있는 상태 값을 갱신하고 원장을 갱신한다.

③ Orderer : Endorsing 피어에 의해 검증된 트랜잭션을 순서화 한다. dApp 클라이언트가 송신한 트랜잭션을 받아서 시간 순으로 정렬한다.

④ Channel : 동일한 원장을 공유하는 하나의 블록체인 네트워크를 구성한다. Peer는 다수의 채널에 참여할 수 있고, 이 때 다수의 블록체인을 관리해야 한다.

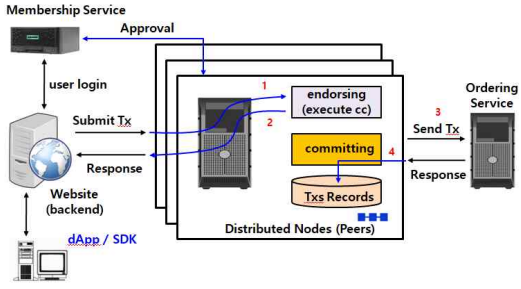

Fig. 2. Elements of HLF

HLF에서 트랜잭션이 처리되는 과정은 다음과 같다[10].
① dApp에서 클라이언트가 HFC SDK를 이용해 트랜잭션을 발생시킨다.
② Endorsing 피어가 트랜잭션을 검증하고 실행한 후 Read/Write Set을 클라이언트에 반환한다.
- 트랜잭션이 형식에 맞게 구성되었는지, 서명이 유효한지 검증
- 원장 업데이트는 진행하지 않는다.
③ 클라이언트는 Endorsing 피어로부터 반환 받은 응답을 Orderer에 전파한다. 트랜잭션에는 Read/Write Set, 서명, 채널 ID 등이 포함되어 있다.
④ Orderer는 트랜잭션을 정렬한다. dApp 클라이언트가 송신한 트랜잭션을 받아서 시간 순으로 정렬한다.
⑤ 트랜잭션 검증 후 커밋(Commit) 한다.
- 트랜잭션은 채널의 모든 (또는 Anchor) 피어에게 전달된다.
- 블록은 채널의 앵커피어를 통하여 모든 피어에게 전파된다.
⑥ 모든 피어에서 상태 값이 갱신되고 원장이 업데이트된다.

3. HLF 기반의 실습프레임워크 구조

3.1 프레임워크 구성요소 및 기능

HLF 서비스를 위하여 필요한 구성 요소는 크게 블록체인 네트워크, 체인코드, dApp으로 분류할 수 있다. 따라서 실습프레임워크은 HLF 네트워크 구성을 지원하는 기능, 체인코드 실행을 지원하는 기능, dApp 실행을 지원하는 기능으로 구성되어 있다.

HLF 네트워크 설정과 구성을 지원하기 위해서 프레임워크에서 수행해야 할 내용과 설정은 다음과 같다.
① 수행해야 할 내용
- 트랜잭션은 비밀키 (KeyStore)로 서명하고 공개키(SignCerts)로 검증하기 위한 키 생성
- 네트워크 구성원들에게 인증서(Certificates) 발급
- 채널 구성정보의 생성
- 네트워크의 채널과 제네시스 블록을 생성하고 앵커피어를 설정
- “docker-compose up” 명령으로 CA, Peer, Orderer, CouchDB 컨테이너 실행
- CLI 컨테이너를 통해 채널을 생성
- 기관의 피어는 생성된 채널에 참여
② 설정 파일 : crypto-config.yaml, configtx.yaml, docker-compose.yml
③ 실행 명령어 : cryptogen, configtxgen, docker-compose
④ 실행 스크립트 파일 : generate.sh, startHLFNetwork.sh
⑤ 생성된 결과물 저장 : config/, crypto-config/

프레임워크에서 체인코드의 빌드, 배포, 활성화, 테스트 기능을 지원하는 기능은 다음과 같다.
① 체인코드는 Go 언어로 작성하고 호스트에서 빌드하여 컨테이너가 동작하는 피어에 배포한다.
② 체인코드 배포를 위해서는 필요한 컨테이너가 네트워크 구성단계에서 동작하고 있어야 한다. CA, Peer, Order, CouchDB 도커 이미지가 컨테이너로 동작하면서 필요한 다양한 구성정보는 환경변수, 작업 디렉토리의 지정, 호스트의 공유볼륨을 통하여 컨테이너에 반영하도록 되어 있다.
③ CLI 컨테이너에서 “peer chaincode install”, “peer chaincode instantiate” 명령을 사용하여 체인코드를 배포하고 활성화 한다. 특히 체인코드 활성화 단계에서 Orderer와 관련되는 정책을 설정할 수 있다.
④ 체인코드가 정상적으로 동작하고 있는지 “peer chaincode invoke”, “peer chaincode query” 명령을 사용하여 확인한다.

dApp은 크게 프런트앤드와 백앤드로 구성되어 있다. 프런트앤드 부분은 기존 웹의 형태로 HTML, CSS, Javascript, jQuery 등으로 구현된다. 백앤드 부분은 프런트앤드와의 연결을 처리하는 기능과 체인코드와 연결되어 수행되는 부분으로 구성되어 있다. 프런트앤드와 통신하기 위한 기능은 기존 웹의 서버 측에서 사용하는 Nodejs, Embedded Javascript (EJS), Express 등을 사용한다[11]. 따라서 dApp 실행을 지원하기 위한 프레임워크의 기능은 다음과 같다.
① Nodejs 설치 확인
② Express, EJS, Fabric-ca-client, Fabric-network 모듈 다운로드를 위한 package.json 파일 구성
③ 관리자 등록 기능
④ 사용자 등록 기능
⑤ dApp 실행 기능

3.2 프레임워크의 동작

실습프레임워크의 동작은 세 부분으로 단계적으로 수행된다.

① HLF 네트워크 구성을 위한 설정 및 동작 수행
② 체인코드의 빌드, 배포, 활성화, 테스트
③ dApp 실행을 위한 환경 설정 및 동작

실습프레임워크는 위 3단계 과정이 순차적으로 실행될 수 있도록 구성되어 있다. 컨소시엄을 구성하여 필요한 서비스를 제공하기 위해서 기관, Peer, Orderer, MSP 등의 개체가 필요하고, 원장을 공유하기 위한 채널구성 등이 매우 복잡하여 효율적인 네트워크 설정이 요구된다. 따라서 HLF 네트워크에서 다양한 기능을 수행하는 각 구성요소를 효율적으로 구성, 설정하고 테스트할 수 있도록 도커 설정파일을 작성하여 컨테이너 기반으로 동작하도록 한다.

HLF 네트워크의 동작은 네트워크 설정, 구성, 실행의 순서로 이루어진다[12]. Table 4에 실행되는 순서와 내용을 보여주고 있다.

Table 4. Framework Functions for HLF Network

Functions	Description
Reset Environment	docker network prune docker stop $(docker ps -aq) docker rm -f $(docker ps -aq)
crypto-config.yaml	script to add Orgx in "Profiles"
configtx.yaml	script to add &Orgx in "Organizations" section script to add *Orgx in "Profiles"
generate.sh - cryptogen - configtxgen	1. set channel Name 2. generate crypto material 3. generate genesis block for orderer 4. generate channel configuration Tx 5. generate anchor peer Tx for each Orgx
Configuration data	config/, crypto-config/
startHLFNetwork.sh	1. docker-compose -f docker-compose.yml up -d ... 2. Create the channel 3. Join peers to the channel
Check HLF network status	docker exec cli peer node status docker exec cli peer channel list

체인코드는 기본적으로 Go 언어 기반으로 작성하고 빌드하여 피어에 배포한다. 컨테이너 기반으로 동작하는 환경에서 연결 포트, 호스트와의 공유 볼륨, 컨테이너 내부에서 사용하기 위한 환경변수 등 다양한 설정을 통하여 체인코드를 배포하고 동작 수행에 따른 결과를 기록하고 채널을 통하여 CouchDB에 상태를 반영하고 원장에는 트랜잭션의 히스토리를 블록으로 추가하게 된다[13]. Table 5에 체인코드와 관련되는 기능을 실행하는 순서와 내용을 보여주고 있다.

Table 5. Framework Functions for Chaincode

Functions	Description
Install chaincode	docker exec -it cli bash peer chaincode install -n mycc -v 1.0 -p github.com/mycc
Instantiate chaincode	peer chaincode instantiate -n mycc -v 1.0 -C paper -c '{...}' -P 'OR (...)'
Chaincode testing	peer chaincode invoke -n mycc -C paper -c '{...}' peer chaincode query -n mycc -C paper -c '{...}'

HLF 네트워크에 체인코드가 배포되어 정상적으로 동작하는 것을 확인한 후, dApp을 통하여 체인코드에서 제공하는 기능을 호출할 수 있다[14]. Table 6에 dApp의 수행과 관련되는 기능을 실행하는 순서와 내용을 보여주고 있다.

Table 6. Framework Functions for dApp

Functions	Description
Setup dApp proj Install nodejs pkg	npm init npm install express, ejs, fabric-ca-client, fabric-network npm install
Register admin	FABRIC_CA_SERVER_CA_KEYFILE node enrollAdmin.js
Register users	node registerUser.js
Query chaincode	node query.js
Invoke chaincode	node invoke.js
Start dApp server	node start.js

4. 실습 프레임워크 구현 및 동작결과

4.1 실습프레임워크 구현

실습프레임워크의 동작은 labFrameWork.js에서 시작된다. labFrameWork.js의 주요골격은 다음과 같다.

```
app.get('/', function (req, res) {
    res.render('index',
        {title: "Main Page", activate: "index"});
});
app.get('/setNetwork', function (req, res) {
    res.render('setNetworkFile',
```

```
        {title: "Construct Network", activate:
"setNetworkMenu"});
});
app.get('/setContract', function (req, res) {
    res.render('setContractFile',
        {title: "Deploy Contract", activate: "setContractMenu"
});
});
app.get('/setdApp', function (req, res) {
    res.render('setdAppFile',
        {title: "Test dApp", activate: "setdAppMenu"});
});
// start labFrameWork
app.listen(8800, "0.0.0.0");
console.log("Framework is Running on
http://127.0.0.1:8800");
```

Fig. 3에 구현된 프레임워크의 초기 동작화면을 보여주고 있다.

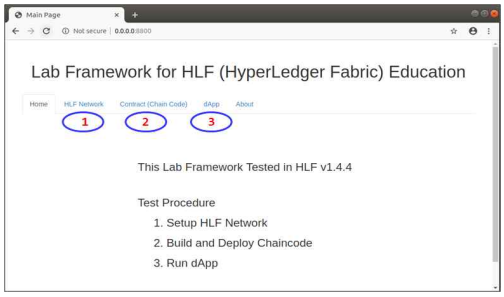

Fig. 3. http://localhost:8800/

실습 프레임워크의 네트워크 구성과 동작을 위한 핵심 기능은 runNetworkScript.jes를 통하여 이루어진다. 코드 골격은 다음과 같다.

```
const {exec} = require('child_process');
var networkScript = exec(
    'sh setNetwork.sh $id',
    (error, stdout, stderr) => {...}
);
```

Fig. 4에 네트워크 테스트를 위한 동작화면을 보여주고 있다.

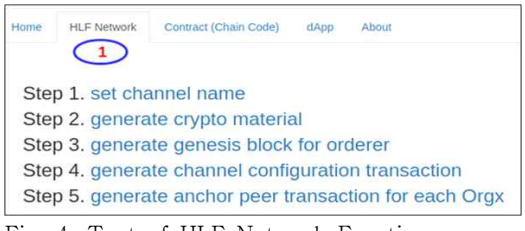

Fig. 4. Test of HLF Network Functions

실습 프레임워크의 체인코드 동작을 위한 핵심 기능은 runContractScript.jes를 통하여 이루어진다. 코드 골격은 다음과 같다.

```
const {exec} = require('child_process');
var ccScript = exec(
    'sh runContractScript.sh $id',
    (error, stdout, stderr) => {...}
);
```

Fig. 5에 체인코드 테스트를 위한 동작화면을 보여주고 있다.

Fig. 5. Test of Chaincode Functions

실습 프레임워크의 dApp 실행을 위한 코드 골격은 다음과 같다.

```
// Connection to Chaincode
const {FileSystemWallet, Gateway} =
 require('fabric-network');
const walletPath = path.join(process.cwd(), 'wallet');
const wallet = new FileSystemWallet(walletPath);
const gateway = new Gateway();
await gateway.connect(... {wallet, identity: 'user1', ...});
const network = await gateway.getNetwork('mychannel');
const contract = network.getContract('mycc');
// Generate transaction
result1 = await contract.submitTransaction('registerStudent',
args);
result2 = await contract.evaluateTransaction('getGrade',
studentName);
// Call Chaincode test through script
app.get('/dAppQueryTest', function (req, res) {
    const {exec} = require('child_process');
    var myscript =
      exec('sh dAppQueryTest.sh', (error, stdout, stderr) =>
        {...});
})
```

Fig. 6에 dApp 테스트를 위한 동작화면을 보여주고 있다.

Fig. 6. Request dApp Execution

4.2 실습프레임워크 동작결과

프레임워크의 동작을 확인하기 위하여 Table 7과 같은 네트워크 환경을 구성한다.

Table 7. Test Network Configuration

Elements	Org Domain (x.example.com)	Anchor Peer	CouchDB
Org	org1.	peer0.org1. port: 7001	couchdb1 port: 7002
	org2.	peer0.org2. port: 8001	couchdb2 port: 8002
	org3.	peer0.org3. port: 9001	couchdb3 port: 9002
orderer	orderer. (port: 7000), OrdererType: Solo		
CA	ca.example.com		
Channel	paper		
CLI	Management CLI container by Administrator		

위와 같은 테스트 환경을 구성하면 9개의 컨테이너가 실행하게 된다. 실습 프레임워크의 동작은 “node labFrameWork.js" 명령을 실행하면 된다. HLF 네트워크를 동작시키고 체인코드를 배포하고 dApp을 단계적으로 실행한다. “docker ps" 명령을 통하여 실행되고 있는 컨테이너를 확인할 수 있다. 이를 통하여 HLF 네트워크가 정상적으로 동작하는 것을 알 수 있다(Fig. 7).

```
root# docker ps -a --format "table {{.ID}}\t{{.Image}}\t{{.Command}}\t{{.Names}}"
CONTAINER ID        IMAGE                          COMMAND                  NAMES
871da2ad944d        hyperledger/fabric-peer        "peer node start"        peer0.org2.example.com
cff1ce51870e        hyperledger/fabric-peer        "peer node start"        peer0.org3.example.com
3e40c9540f9f        hyperledger/fabric-peer        "peer node start"        peer0.org1.example.com
fa61088fd7a1        hyperledger/fabric-couchdb     "tini -- /docker-ent…"   couchdb2
671ef8496ac         hyperledger/fabric-couchdb     "tini -- /docker-ent…"   couchdb1
82c465668d0d        hyperledger/fabric-couchdb     "tini -- /docker-ent…"   couchdb3
e5480d90f19         hyperledger/fabric-orderer     "orderer"                orderer.example.com
3eec5ad9c66b        hyperledger/fabric-tools       "/bin/bash"              cli
a4eb610ff236        hyperledger/fabric-ca          "sh -c 'fabric-ca-se…"   ca.example.com
root#
```

Fig. 7. Check Running Containers

dApp은 체인코드를 통하여 블록체인에 데이터를 저장하고 조회할 수 있다. 테스트를 위하여 구현된 체인코드는 성적관리시스템으로 아래의 기능을 제공한다.

① 학생 이름을 등록한다(Fig. 8).

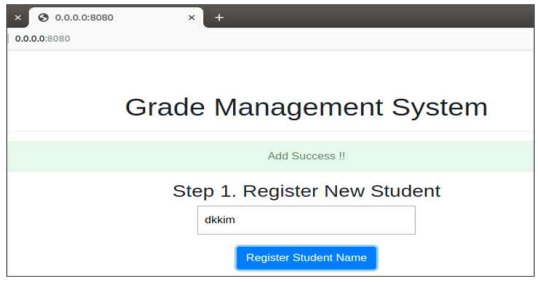

Fig. 8. Register Student Name

② 학생에 대하여 교과목명과 성적을 저장한다(Fig. 9).

Fig. 9. Save Course Name and Grade

③ 학생을 조회하면 저장된 교과목의 성적과 평균을 출력한다(Fig. 10).

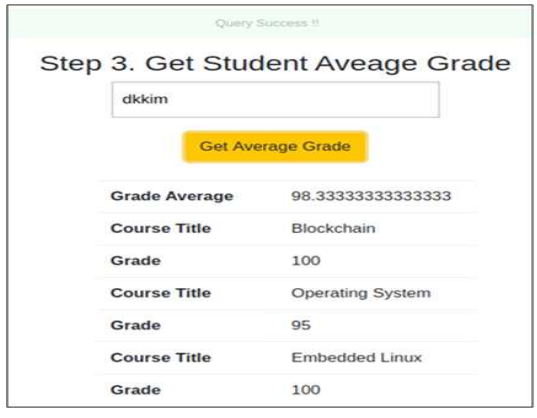

Fig. 10. Query Student Average Grade

5. 결론

HLF 네트워크는 Endorsing Peer, Anchor Peer,

Committing Peer, Orderer, MSP, Channel 등의 요소로 구성된다. 따라서 효율적인 실습을 위해서는 각 요소에 대한 체계적인 이해가 필요하다. 특히 클라우드 기반에서 동작하는 블록체인 네트워크는 일반적으로 도커 컨테이너 기반으로 실행되기 때문에 이에 대한 설정 또한 HLF 네트워크 구성에서 매우 중요하다. 본 논문에서는 이와 같이 복잡한 구성요소의 기능과 동작을 효율적이고 체계적으로 이해할 수 있는 실습프레임워크를 설계하고 구현하였다. 이것을 통하여 네트워크 구성과 컨테이너 실행과정, 체인코드의 배포 및, 활성화 과정, dApp에 의한 트랜잭션 생성과 체인코드 함수 실행과정을 단계적으로 테스트 하였다. 그 결과 실습 프레임워크를 통하여 블록체인 서비스를 위한 전체 흐름을 이해하는 것이 매우 용이하였고, 향후 네트워크의 확장 등에도 전체 흐름에 대한 체계적인 이해가 가능할 것으로 판단되었다.

REFERENCES

[1] Satoshi Nakamoto. (2008). Bitcoin: A Peer to Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[2] Renato P. dos Santos. (2018). PoW, PoS, & Hybrid protocols: A Matter of Complexity. https://arxiv.org/ftp/arxiv/papers/1805/1805.08674.pdf

[3] Mattumd. (2019). Running and Testing Smart Contracts for Hyperledger Fabric. Retrieved from https://developer.ibm.com/recipes/tutorials/running-and-testing-smart-contracts-for-hyperledger-fabric/

[4] Gilad Haimov. (2018). How to Create an ERC20 Token the Simple Way. https://www.toptal.com/ethereum/create-erc20-token-tutorial

[5] Gaurav Agrawal. (2019). ERC-721 Token. https://medium.com/quiknode/erc-721-token-ea80c7195102

[6] Luit Hollander. (2019). Ethereum Virtual Machine. https://medium.com/mycryppto/the-ethereum-virtual-machine-how-does-it-work-9abac2b7c9e

[7] Shuai Wang et al. (2018). *An Ove rview of Smart Contract: Architecture, Applica tions, and Future Trends.* IEEE Intelligent Vehi cles Symposium.

[8] Elli Androulaki et al. (2018). *Hyperledger Fabric: A Distributed Operating S ystem for Permissioned Blockchains.* Proceedings of the Thirteenth EuroSys Conference.

[9] Christian Cachin. (2016). *Architecture of the Hyperledger Blockchain Fabric.* Workshop on Distributed cryptocurrencies.

[10] M Vukolic. (2017). *Rethinking permissioned blockchains.* Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts.

[11] Jennifer Bland. (2019). Example using EJS Templating in Node.js Application. https://www.jenniferbland.com/example-using-ejs-templating-in-node-js-application/

[12] K. C. Tam. (2018). Understanding First Network Example in Hyperledger Fabric. https://kctheservant.medium.com/understanding-first-network-example-in-hyperledger-fabric-part-1-c03391af798

[13] Mattumd. (2019). Writing Hyperledger Fabric Chaincode Using Go Programming Language. https://developer.ibm.com/recipes/tutorials/writing-hyperledger-fabric-chaincode-using-go-programming-language/

[14] Coral Health. (2018). Build a DApp on Hyperledger the Easy Way. https://mycoralhealth.medium.com/build-a-dapp-on-hyper ledger-the-easy-way-178c39e503fa

김 도 규(Do-Kyu Kim) [정회원]

· 1987년 2월: 경북대학교 전자공학과
· 1989년 2월: KAIST 전기전자공학과(공학석사)
· 1993년 2월: KAIST 전기전자공학과(공학박사)
· 1995년 2월: 정보통신부통신사무관
· 1995년 3월~현재: 성결대학교 정보통신공학과 교수
· 관심분야 : Blockchain, Embedded System, Network
· E-Mail : dkkim@sungkyul.edu