연구 보고서

작성자	김한호	작성일자	2021.05.09.

1. 연구 계획

- asset-transfer-private-data를 원본으로 프로젝트 go언어 체인코드 구현

2. 논문 연구 진행

```
h2kim@h2kim-VBox:~/fabric-samples/e2-network$ ./network.sh deployCC -ccn e2-list-basic -ccp ../e2
-list-basic/chaincode-go/ -ccl go
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: e2-list-basic
- CC_SRC_PATH: ../e2-list-basic/chaincode-go/
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
Vendoring Go dependencies at ../e2-list-basic/chaincode-go/
~/fabric-samples/e2-list-basic/chaincode-go ~/fabric-samples/e2-network
~/fabric-samples/e2-network
Finished vendoring Go dependencies
Using organization 1
+ peer lifecycle chaincode package e2-list-basic.tar.gz --path ../e2-list-basic/chaincode-go/ --l
ang golang --label e2-list-basic_1.0
+ res=0
Chaincode is packaged on peer0.org1
Installing chaincode on peer0.org1...
Using organization 1
+ peer lifecycle chaincode install e2-list-basic.tar.gz
+ res=0
2021-05-05 14:29:20.994 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed
 remotely: response:<status:200 payload:"\nRe2-list-basic_1.0:09d85b99f311c9970bf1c26ae5f77093606
7615887a2e3d149c8263e9cf6822f\022\021e2-list-basic_1.0" >
2021-05-05 14:29:20.994 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode
 code package identifier: e2-list-basic_1.0:09d85b99f311c9970bf1c26ae5f770936067615887a2e3d149c82
63e9cf6822f
Chaincode is installed on peer0.org1
Install chaincode on peer0.org2...
Using organization 2
+ peer lifecycle chaincode install e2-list-basic.tar.gz
```

설정한 네트워크에 체인코드를 올려서 제대로 작동하는지 확인함. 2021.05.02. 연구에서 작성한 공개키 암호화 시스템을 이용한 암호화 복호화 테스트를 하는 체인코드를 사용하여 네트워크에 올리는 작업을 수행함.

```
Query installed successful on peer0.org1 on channel
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer
.example.com --tls --cafile /home/h2kim/fabric-samples/e2-network/organizations/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelI
D mychannel --name e2-list-basic --version 1.0 --package-id e2-list-basic_1.0:09d85b99f311c9970bf
1c26ae5f770936067615887a2e3d149c8263e9cf6822f --sequence 1
+ res=0
2021-05-05 14:29:46.210 KST [chaincodeCmd] ClientWait -> INFO 001 txid [5beddda10e485754de41cdb4f
aa08bd7f7e6b12473cc3a3e107d55423015ecd9] committed with status (VALID) at
Chaincode definition approved on peer0.org1 on channel 'mychannel'
Using organization 1
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3
 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name e2-list-basic --vers
ion 1.0 --sequence 1 --output json
+ res=0
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3
 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name e2-list-basic --vers
ion 1.0 --sequence 1 --output json
+ res=0
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3
 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name e2-list-basic --vers
ion 1.0 --sequence 1 --output json
+ res=0
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3
 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name e2-list-basic --vers
ion 1.0 --sequence 1 --output json
+ res=0
{
        "approvals": {
                "civilMSP": false,
                "controlMSP": true,
                "shopMSP": false
        }
}
After 5 attempts, Check commit readiness result on peer0.org1 is INVALID!
Deploying chaincode failed
```

체인코드를 커밋하고 각 피어에 요청을 받지 못하는 것을 확인함.

```
319 ## package the chaincode
320 packageChaincode 1
321
322 ## Install chaincode on peer0.org1 and peer0.org2
323 infoln "Installing chaincode on peer0.org1..."
324 installChaincode 1
325 infoln "Install chaincode on peer0.org2..."
326 installChaincode 2
327
328 ## query whether the chaincode is installed
329 queryInstalled 1
330
331 ## approve the definition for org1
332 approveForMyOrg 1
333
334 ## check whether the chaincode definition is ready to be committed
335 ## expect org1 to have approved and org2 not to
336 checkCommitReadiness 1 "\"Org1MSP\": true" "\"Org2MSP\": false"
337 checkCommitReadiness 2 "\"Org1MSP\": true" "\"Org2MSP\": false"
338
339 ## now approve also for org2
340 approveForMyOrg 2
341
342 ## check whether the chaincode definition is ready to be committed
343 ## expect them both to have approved
344 checkCommitReadiness 1 "\"Org1MSP\": true" "\"Org2MSP\": true"
345 checkCommitReadiness 2 "\"Org1MSP\": true" "\"Org2MSP\": true"
346
347 ## now that we know for sure both orgs have approved, commit the definition
348 commitChaincodeDefinition 1 2
349
350 ## query on both orgs to see that the definition committed successfully
351 queryCommitted 1
352 queryCommitted 2
353
354 ## Invoke the chaincode - this does require that the chaincode have the 'initLedger'
355 ## method defined
356 if [ "$CC_INIT_FCN" = "NA" ]; then
357   infoln "Chaincode initialization is not required"
358 else
359   chaincodeInvokeInit 1 2
360 fi
361
362 exit 0
```

deploy.cc 쉘스크립트 파일에서 피어에 변경에 따른 설정이 제대로 작성되지 않은 것을 확인함.

```
## Install chaincode on peer0.controlMSP, peer0.civilMSP and peer0.shopMSP
infoln "Installing chaincode on peer0.controlMSP..."
installChaincode 1
infoln "Install chaincode on peer0.civilMSP..."
installChaincode 2
infoln "Install chaincode on peer0.shopMSP..."
installChaincode 3

## query whether the chaincode is installed
queryInstalled 1

## approve the definition for control
approveForMyOrg 1

## check whether the chaincode definition is ready to be committed
## expect control to have approved and civil, shop not to
checkCommitReadiness 1 "\"controlMSP\": true" "\"civilMSP\": false" "\"shopMSP\": false"
checkCommitReadiness 2 "\"controlMSP\": true" "\"civilMSP\": false" "\"shopMSP\": false"
checkCommitReadiness 3 "\"controlMSP\": true" "\"civilMSP\": false" "\"shopMSP\": false"

## now approve also for civil
approveForMyOrg 2

## check whether the chaincode definition is ready to be committed
## expect them both to have approved not shop
checkCommitReadiness 1 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": false"
checkCommitReadiness 2 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": false"
checkCommitReadiness 3 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": false"

## now approve also for civil
approveForMyOrg 3

## check whether the chaincode definition is ready to be committed
## expect them both to have approved not shop
checkCommitReadiness 1 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": true"
checkCommitReadiness 2 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": true"
checkCommitReadiness 3 "\"controlMSP\": true" "\"civilMSP\": true" "\"shopMSP\": true"


## now that we know for sure both orgs have approved, commit the definition
commitChaincodeDefinition 1 2 3

## query on both orgs to see that the definition committed successfully
queryCommitted 1
queryCommitted 2
queryCommitted 3
```

변경한 네트워크에 따라서 deployCC 쉘스크립트를 변경함.

```
Chaincode is installed on peer0.org2
Install chaincode on peer0.shopMSP...
Using organization 3
+ peer lifecycle chaincode install e2-list-basic.tar.gz
+ res=0
2021-05-05 15:05:14.931 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed
 remotely: response:<status:200 payload:"\nRe2-list-basic_1.0:09d85b99f311c9970bf1c26ae5f77093606
7615887a2e3d149c8263e9cf6822f\022\021e2-list-basic_1.0" >
2021-05-05 15:05:14.931 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode
 code package identifier: e2-list-basic_1.0:09d85b99f311c9970bf1c26ae5f770936067615887a2e3d149c82
63e9cf6822f
Chaincode is installed on peer0.org3
Using organization 1
+ peer lifecycle chaincode queryinstalled
+ res=0
Installed chaincodes on peer:
Package ID: e2-list-basic_1.0:09d85b99f311c9970bf1c26ae5f770936067615887a2e3d149c8263e9cf6822f, L
abel: e2-list-basic_1.0
Query installed successful on peer0.org1 on channel
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer
.example.com --tls --cafile /home/h2kim/fabric-samples/e2-network/organizations/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelI
D mychannel --name e2-list-basic --version 1.0 --package-id e2-list-basic_1.0:09d85b99f311c9970bf
1c26ae5f770936067615887a2e3d149c8263e9cf6822f --sequence 1
+ res=0
2021-05-05 15:05:17.451 KST [chaincodeCmd] ClientWait -> INFO 001 txid [6e3382f05733e40b072a94e49
b2355811249cab78ab866931b6bdd2397c4c6ad] committed with status (VALID) at
Chaincode definition approved on peer0.org1 on channel 'mychannel'
Using organization 1
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3
 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name e2-list-basic --vers
ion 1.0 --sequence 1 --output json
+ res=0
{
        "approvals": {
                "civilMSP": false,
                "controlMSP": true,
                "shopMSP": false
        }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'my
channel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org2 on channel 'mychannel'...
```

위에서 작동이 실패했던 $ ./network.sh deployCC -ccn e2-list-basic -ccp ../e2-list-basic/chaincode-go/ -ccl go 명령어를 실행함. 오류가 발생했던 곳에서 다음 단계로 넘어가는 것을 확인할 수 있음.

```
Flags:
      --channel-config-policy string   The endorsement policy associated to this chaincode specif
ied as a channel config policy reference
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
ding the file name
      --connectionProfile string       The fully qualified path to the connection profile that pr
ovides the necessary connection information for the network. Note: currently only supported for p
roviding peer connection information
  -E, --endorsement-plugin string      The name of the endorsement plugin to be used for this cha
incode
  -h, --help                           help for commit
      --init-required                  Whether the chaincode requires invoking 'init'
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
annel
      --signature-policy string        The endorsement policy associated to this chaincode specif
ied as a signature policy
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
 the peers to connect to. The order and number of certs specified should match the --peerAddresse
s flag
  -V, --validation-plugin string       The name of the validation plugin to be used for this chai
ncode
  -v, --version string                 Version of the chaincode
      --waitForEvent                   Whether to wait for the event from each peer's deliver fil
tered service signifying that the transaction has been committed successfully (default true)
      --waitForEventTimeout duration   Time to wait for the event from each peer's deliver filter
ed service signifying that the 'invoke' transaction has been committed successfully (default 30s)

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
te(s) for the ordering endpoint
      --certfile string                     Path to file containing PEM-encoded X509 public key t
o use for mutual TLS communication with the orderer endpoint
      --clientauth                          Use mutual TLS when communicating with the orderer en
dpoint
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
e for mutual TLS communication with the orderer endpoint
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
connection to the orderer.
      --tls                                 Use TLS when communicating with the orderer endpoint

Chaincode definition commit failed on peer0.org3 on channel 'mychannel' failed
Deploying chaincode failed
```

명령 수행을 한 이후에 제대로 커밋이 되지 않고 체인코드 배포가 실패하는 것을 확인할 수 있었음.

```
Checking the commit readiness of the chaincode definition successful on peer0.org3 on channel 'my
channel'
Using organization 1
Using organization 2
Using organization 3
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.
com --tls --cafile /home/h2kim/fabric-samples/e2-network/organizations/ordererOrganizations/examp
le.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychann
el --name e2-list-basic --peerAddresses localhost:7051 --tlsRootCertFiles --peerAddresses localho
st:8051 --tlsRootCertFiles --peerAddresses localhost:10051 --tlsRootCertFiles --version 1.0 --seq
uence 1
+ res=1
Error: failed to validate peer connection parameters: number of peer addresses (1) does not match
 the number of TLS root cert files (3)
Usage:
  peer lifecycle chaincode commit [flags]

Flags:
      --channel-config-policy string   The endorsement policy associated to this chaincode specif
ied as a channel config policy reference
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
ding the file name
      --connectionProfile string       The fully qualified path to the connection profile that pr
ovides the necessary connection information for the network. Note: currently only supported for p
roviding peer connection information
  -E, --endorsement-plugin string      The name of the endorsement plugin to be used for this cha
incode
  -h, --help                           help for commit
      --init-required                  Whether the chaincode requires invoking 'init'
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
annel
      --signature-policy string        The endorsement policy associated to this chaincode specif
ied as a signature policy
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
 the peers to connect to. The order and number of certs specified should match the --peerAddresse
s flag
  -V, --validation-plugin string       The name of the validation plugin to be used for this chai
ncode
  -v, --version string                 Version of the chaincode
      --waitForEvent                   Whether to wait for the event from each peer's deliver fil
tered service signifying that the transaction has been committed successfully (default true)
      --waitForEventTimeout duration   Time to wait for the event from each peer's deliver filter
ed service signifying that the 'invoke' transaction has been committed successfully (default 30s)
```

체인코드 커밋을 할 때 –tlsRootCertFiles 뒤에 tlsRootCertFiles의 위치가 들어가지 않은 것을 확인할 수 있었음.

```
Using organization 3
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.
com --tls --cafile /home/h2kim/fabric-samples/e2-network/organizations/ordererOrganizations/examp
le.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychann
el --name e2-list-basic --peerAddresses localhost:7051 --tlsRootCertFiles /home/h2kim/fabric-samp
les/e2-network/organizations/peerOrganizations/control.example.com/peers/peer0.control.example.co
m/tls/ca.crt --peerAddresses localhost:8051 --tlsRootCertFiles /home/h2kim/fabric-samples/e2-netw
ork/organizations/peerOrganizations/civil.example.com/peers/peer0.civil.example.com/tls/ca.crt --
peerAddresses localhost:10051 --tlsRootCertFiles /home/h2kim/fabric-samples/e2-network/organizati
ons/peerOrganizations/shop.example.com/peers/peer0.shop.example.com/tls/ca.crt --version 1.0 --se
quence 1
+ res=0
2021-05-05 15:16:34.285 KST [chaincodeCmd] ClientWait -> INFO 001 txid [19c5a252a857026fa8bda416f
5c59f12c248922fd667f7373ebaecbef227bde3] committed with status (VALID) at localhost:7051
2021-05-05 15:16:34.410 KST [chaincodeCmd] ClientWait -> INFO 002 txid [19c5a252a857026fa8bda416f
5c59f12c248922fd667f7373ebaecbef227bde3] committed with status (VALID) at localhost:8051
2021-05-05 15:16:34.467 KST [chaincodeCmd] ClientWait -> INFO 003 txid [19c5a252a857026fa8bda416f
5c59f12c248922fd667f7373ebaecbef227bde3] committed with status (VALID) at localhost:10051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name e2-list-basic
+ res=0
Committed chaincode definition for chaincode 'e2-list-basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [civilMS
P: true, controlMSP: true, shopMSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name e2-list-basic
+ res=0
Committed chaincode definition for chaincode 'e2-list-basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [civilMS
P: true, controlMSP: true, shopMSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Using organization 3
Querying chaincode definition on peer0.org3 on channel 'mychannel'...
Attempting to Query committed status on peer0.org3, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name e2-list-basic
+ res=0
Committed chaincode definition for chaincode 'e2-list-basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [civilMS
P: true, controlMSP: true, shopMSP: true]
Query chaincode definition successful on peer0.org3 on channel 'mychannel'
Chaincode initialization is not required
```

envVar 쉘스크립트 파일에 값이 입력될 수 있도록 변경 실행함. 제대로 파일의 경로가 들어가는 것을 확인할 수 있었음.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-network$ docker network ls
NETWORK ID          NAME                                DRIVER              SCOPE
151148a358fb        2org1peercouchdbsoloraft_default    bridge              local
69cb8b9d6cbe        bridge                              bridge              local
60f71b194ad1        host                                host                local
a88ea70b5f1b        net_e2                              bridge              local
2587366efe76        net_fabric                          bridge              local
fa31556decad        none                                null                local
h2kim@h2kim-VBox:~/fabric-samples/e2-network$
```

도커에서 제대로 설정한 네트워크가 돌아가고 있는지 확인함. 응용프로그램 작동할 때에 네트워크 상에 이상이 있는지 확인하여 차후에 코드 부분에서 문제가 발생했을 때 네트워크를 제외하기 위함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-basic/application-javascript$ node app.js

--> Fabric client user & Gateway init: Using Org1 identity to Org1 Peer
Error in transaction: Error: no such file or directory: /home/h2kim/fabric-samples/test-network/o
rganizations/peerOrganizations/org1.example.com/connection-org1.json
Error: no such file or directory: /home/h2kim/fabric-samples/test-network/organizations/peerOrgan
izations/org1.example.com/connection-org1.json
    at exports.buildCCPOrg1 (/home/h2kim/fabric-samples/test-application/javascript/AppUtil.js:17
:9)
    at initContractFromOrg1Identity (/home/h2kim/fabric-samples/e2-list-basic/application-javascr
ipt/app.js:40:21)
    at main (/home/h2kim/fabric-samples/e2-list-basic/application-javascript/app.js:118:35)
    at Object.<anonymous> (/home/h2kim/fabric-samples/e2-list-basic/application-javascript/app.js
:175:1)
    at Module._compile (internal/modules/cjs/loader.js:778:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:789:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:831:12)
```

node app.js를 이용해 프로그램을 실행하게 되니 연결할 수 있는 정보를 찾을 수 없다고 에러를 발생함.

```
 1 /*
 2  * Copyright IBM Corp. All Rights Reserved.
 3  *
 4  * SPDX-License-Identifier: Apache-2.0
 5  */
 6
 7 'use strict';
 8
 9 const fs = require('fs');
10 const path = require('path');
11
12 exports.buildCCPOrg1 = () => {
13     // load the common connection configuration file
14     const ccpPath = path.resolve(__dirname, '..', '..', 'test-network', 'organizations', 'pee
   rOrganizations', 'org1.example.com', 'connection-org1.json');
15     const fileExists = fs.existsSync(ccpPath);
16     if (!fileExists) {
17         throw new Error(`no such file or directory: ${ccpPath}`);
18     }
19     const contents = fs.readFileSync(ccpPath, 'utf8');
20
21     // build a JSON object from the file contents
22     const ccp = JSON.parse(contents);
23
24     console.log(`Loaded the network configuration located at ${ccpPath}`);
25     return ccp;
26 };
27
28 exports.buildCCPOrg2 = () => {
29     // load the common connection configuration file
30     const ccpPath = path.resolve(__dirname, '..', '..', 'test-network',
31         'organizations', 'peerOrganizations', 'org2.example.com', 'connection-org2.json');
32     const fileExists = fs.existsSync(ccpPath);
33     if (!fileExists) {
34         throw new Error(`no such file or directory: ${ccpPath}`);
35     }
36     const contents = fs.readFileSync(ccpPath, 'utf8');
37
38     // build a JSON object from the file contents
39     const ccp = JSON.parse(contents);
40
41     console.log(`Loaded the network configuration located at ${ccpPath}`);
42     return ccp;
43 };
44
45 exports.buildWallet = async (Wallets, walletPath) => {
46     // Create a new  wallet : Note that wallet is for managing identities.
```

const ccpPath에서 연결하는 파일을 설정이 변경되지 않음을 확인함.

```
/*
 * Copyright IBM Corp. All Rights Reserved.
 *
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const fs = require('fs');
const path = require('path');

exports.buildCCPOrg1 = () => {
    // load the common connection configuration file
    const ccpPath = path.resolve(__dirname, '..', '..', 'e2-network', 'organizations', 'peerOrganizations', 'control.example.com', 'connection-control.json');
    const fileExists = fs.existsSync(ccpPath);
    if (!fileExists) {
        throw new Error(`no such file or directory: ${ccpPath}`);
    }
    const contents = fs.readFileSync(ccpPath, 'utf8');

    // build a JSON object from the file contents
    const ccp = JSON.parse(contents);

    console.log(`Loaded the network configuration located at ${ccpPath}`);
    return ccp;
};

exports.buildCCPOrg2 = () => {
    // load the common connection configuration file
    const ccpPath = path.resolve(__dirname, '..', '..', 'e2-network',
        'organizations', 'peerOrganizations', 'civil.example.com', 'connection-civil.json');
    const fileExists = fs.existsSync(ccpPath);
    if (!fileExists) {
        throw new Error(`no such file or directory: ${ccpPath}`);
    }
    const contents = fs.readFileSync(ccpPath, 'utf8');

    // build a JSON object from the file contents
    const ccp = JSON.parse(contents);

    console.log(`Loaded the network configuration located at ${ccpPath}`);
    return ccp;
};

exports.buildWallet = async (Wallets, walletPath) => {
    // Create a new  wallet : Note that wallet is for managing identities.
```

변경된 네트워크에 맞춰서 설정을 변경함.

```
const { buildCAClient, registerAndEnrollUser, enrollAdmin } = require('../../test-application/javascript/CAUtil.js');
const { buildCCPOrg1, buildCCPOrg2, buildWallet } = require('../../test-application/javascript/E2Util.js');
```

코드에서 사용하던 AppUtil.js파일을 E2Util.js로 새로 만들어 코드에 적용함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/application-javascript$ node app.js

--> Fabric client user & Gateway init: Using Org1 identity to Org1 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/e2-network/organizations/p
eerOrganizations/control.example.com/connection-control.json
Error in transaction: TypeError: Cannot read property 'tlsCACerts' of undefined
TypeError: Cannot read property 'tlsCACerts' of undefined
    at exports.buildCAClient (/home/h2kim/fabric-samples/test-application/javascript/CAUtil.js:20
:30)
    at initContractFromOrg1Identity (/home/h2kim/fabric-samples/e2-list-private-data/application-
javascript/app.js:51:26)
    at main (/home/h2kim/fabric-samples/e2-list-private-data/application-javascript/app.js:125:35
)
    at Object.<anonymous> (/home/h2kim/fabric-samples/e2-list-private-data/application-javascript
/app.js:384:1)
    at Module._compile (internal/modules/cjs/loader.js:778:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:789:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:831:12)
```

프로그램 실행명령 후 네트워크와 연결하는 것은 작동하지만 tlsCACerts 변수가 정의되지 않아서 읽어올 수 없다고 에러를 발생함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-basic/application-javascript$ node app.js

--> Fabric client user & Gateway init: Using Org1 identity to Org1 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/e2-network/organizations/p
eerOrganizations/control.example.com/connection-control.json
caTLSCACerts:  [ '-----BEGIN CERTIFICATE-----\nMIICXTCCAgOgAwIBAgIQfaM4391/qLPBsnAYfVOFIDAKBggqhk
jOPQQDAjB5MQsw\nCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNU2FuIEZy\nYW5jaXNjbzEcMBo
GA1UEChMTY29udHJvbC5leGFtcGxlLmNvbTEfMB0GA1UEAxMW\nY2EuY29udHJvbC5leGFtcGxlLmNvbTAeFw0yMTA1MDUwNj
A5MDBaFw0zMTA1MDMw\nNjA5MDBaMHkxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYD\nVQQHEw1TYW4
gRnJhbmNpc2NvMRwwGgYDVQQKExNjb250cm9sLmV4YW1wbGUuY29t\nMR8wHQYDVQQDExZjYS5jb250cm9sLmV4YW1wbGUuY2
9tMFkwEwYHKoZIzj0CAQYI\nKoZIzj0DAQcDQgAE3raLXHWEIDq9QLRmixyMCsvrsXOJbG8Am8rnsQnSCtvvFpmy\nEFRbkLB
Ht/IrmgFhNY+Zz3LlpXF7XQeSC8qjsaNtMGswDgYDVR0PAQH/BAQDAgGm\nMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQ
cDATAPBgNVHRMBAf8EBTADAQH/\nMCkGA1UdDgQiBCAl1iOUQmV+Z8rhs9ueL7IZjGawMAIZaHIhjdKpdk1kFjAKBggq\nhkj
OPQQDAgNIADBFAiEA9qjQ3XpGqdM9Vh9Vt/QkyaSMHhhRz/TRWkBs7GDwDDQC\nIALbz+IIO4HPhu8Su6qZgt4sT/QuIRDkKu
Fx0HcGLnjm\n-----END CERTIFICATE-----\n' ]
Built a CA Client named ca-control
Built a file system wallet at /home/h2kim/fabric-samples/e2-list-basic/application-javascript/wal
let/control
2021-05-05T06:50:17.311Z - error: [FabricCAClientService.js]: Failed to enroll admin, error:%o me
ssage=Calling enroll endpoint failed with error [Error: connect ECONNREFUSED 127.0.0.1:7054], sta
ck=Error: Calling enroll endpoint failed with error [Error: connect ECONNREFUSED 127.0.0.1:7054]
    at ClientRequest.request.on (/home/h2kim/fabric-samples/e2-list-basic/application-javascript/
node_modules/fabric-ca-client/lib/FabricCAClient.js:327:19)
    at ClientRequest.emit (events.js:198:13)
    at TLSSocket.socketErrorListener (_http_client.js:401:9)
    at TLSSocket.emit (events.js:198:13)
    at emitErrorNT (internal/streams/destroy.js:91:8)
    at emitErrorAndCloseNT (internal/streams/destroy.js:59:3)
    at process._tickCallback (internal/process/next_tick.js:63:19), errno=ECONNREFUSED, code=ECON
NREFUSED, syscall=connect, address=127.0.0.1, port=7054
Failed to enroll admin user : Error: Calling enroll endpoint failed with error [Error: connect EC
ONNREFUSED 127.0.0.1:7054]
An identity for the admin user does not exist in the wallet
Enroll the admin user before retrying
Error in connecting to gateway: Error: Identity not found in wallet: appUser1
```

console.log 함수를 이용해서 명령어를 작동하고 제대로 불러오는 값이 있다는 것을 확인함. 연결이 제대로 되지 않는 것을 확인함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-network$ ./network.sh up createChannel -s couchdb -ca
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds a
nd using database 'couchdb with crypto from 'Certificate Authorities'
Bringing up network
LOCAL_VERSION=2.2.0
DOCKER_IMAGE_VERSION=2.2.0
CA_LOCAL_VERSION=1.4.8
CA_DOCKER_IMAGE_VERSION=1.4.8
Generate certificates using Fabric CA's
Creating network "net_e2" with the default driver
Creating ca_control ...
Creating ca_civil ...
Creating ca_shop ...
Creating ca_orderer ...
Creating ca_control
Creating ca_civil
Creating ca_shop
Creating ca_civil ... done
Create control Identities
Enroll the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-control --tls.certf
iles /home/h2kim/fabric-samples/e2-network/organizations/fabric-ca/control/tls-cert.pem
2021/05/05 15:51:57 [INFO] Created a default configuration file at /home/h2kim/fabric-samples/e2-
network/organizations/peerOrganizations/control.example.com/fabric-ca-client-config.yaml
2021/05/05 15:51:57 [INFO] TLS Enabled
2021/05/05 15:51:57 [INFO] generating key: &{A:ecdsa S:256}
2021/05/05 15:51:57 [INFO] encoded CSR
2021/05/05 15:51:57 [INFO] Stored client certificate at /home/h2kim/fabric-samples/e2-network/org
anizations/peerOrganizations/control.example.com/msp/signcerts/cert.pem
2021/05/05 15:51:57 [INFO] Stored root CA certificate at /home/h2kim/fabric-samples/e2-network/or
ganizations/peerOrganizations/control.example.com/msp/cacerts/localhost-7054-ca-control.pem
2021/05/05 15:51:57 [INFO] Stored Issuer public key at /home/h2kim/fabric-samples/e2-network/orga
nizations/peerOrganizations/control.example.com/msp/IssuerPublicKey
2021/05/05 15:51:57 [INFO] Stored Issuer revocation public key at /home/h2kim/fabric-samples/e2-n
etwork/organizations/peerOrganizations/control.example.com/msp/IssuerRevocationPublicKey
Register peer0
+ fabric-ca-client register --caname ca-control --id.name peer0 --id.secret peer0pw --id.type pee
r --tls.certfiles /home/h2kim/fabric-samples/e2-network/organizations/fabric-ca/control/tls-cert.
pem
2021/05/05 15:51:57 [INFO] Configuration file location: /home/h2kim/fabric-samples/e2-network/org
anizations/peerOrganizations/control.example.com/fabric-ca-client-config.yaml
2021/05/05 15:51:57 [INFO] TLS Enabled
2021/05/05 15:51:57 [INFO] TLS Enabled
Password: peer0pw
Register user
```

ca연결이 되지 않는 것이라고 판단해서 네트워크를 생성할 때에 -ca명령어를 넣어서 ca를 생성하도록 하여 연결이 될 것으로 예측하고 실행함.

```
60      const appUser1Identity = await walletOrg1.get(Org1UserId);
61      if (!appUser1Identity) {
62          await registerAndEnrollUser(caOrg1Client, walletOrg1, mspOrg1, Org1UserId, 'control.d
    epartment1');
63      }
64
65      try {
66          // Create a new gateway for connecting to Org's peer node.
67          const gatewayOrg1 = new Gateway();
68          //connect using Discovery enabled
69          await gatewayOrg1.connect(ccpOrg1,
70              { wallet: walletOrg1, identity: Org1UserId, discovery: { enabled: true, asLocalho
    st: true } });
71
72          return gatewayOrg1;
73      } catch (error) {
74          console.error(`Error in connecting to gateway: ${error}`);
75          process.exit(1);
76      }
77  }
78
79  async function initContractFromOrg2Identity() {
80      console.log('\n--> Fabric client user & Gateway init: Using Org2 identity to Org2 Peer');
81      const ccpOrg2 = buildCCPOrg2();
82      const caOrg2Client = buildCAClient(FabricCAServices, ccpOrg2, 'ca.civil.example.com');
83
84      const walletPathOrg2 = path.join(__dirname, 'wallet/civil');
85      const walletOrg2 = await buildWallet(Wallets, walletPathOrg2);
86
87      const caOrg2Admin = await walletOrg2.get(caOrg2Client);
88      if(!caOrg2Admin) {
89          await enrollAdmin(caOrg2Client, walletOrg2, mspOrg2);
90      }
91
92      const appUser2Identity = await walletOrg2.get(Org2UserId);
93      if (!appUser2Identity) {
94          await registerAndEnrollUser(caOrg2Client, walletOrg2, mspOrg2, Org2UserId, 'civil.dep
    artment1');
95      }
96
97      try {
98          // Create a new gateway for connecting to Org's peer node.
99          const gatewayOrg2 = new Gateway();
100         await gatewayOrg2.connect(ccpOrg2,
101             { wallet: walletOrg2, identity: Org2UserId, discovery: { enabled: true, asLocalho
    st: true } });
```

ca를 연결할 때 사용되는 buildCAClient의 값을 변경한 ca 값으로 넣어주어 명령을 실행하였음.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-basic/application-javascript$ node app.js

--> Fabric client user & Gateway init: Using Org1 identity to Org1 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/e2-network/organizations/p
eerOrganizations/control.example.com/connection-control.json
Built a CA Client named ca-control
Built a file system wallet at /home/h2kim/fabric-samples/e2-list-basic/application-javascript/wal
let/control
An identity for the admin user already exists in the wallet
Successfully registered and enrolled user appUser1 and imported it into the wallet

--> Fabric client user & Gateway init: Using Org2 identity to Org2 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/e2-network/organizations/p
eerOrganizations/civil.example.com/connection-civil.json
Built a CA Client named ca-civil
Built a file system wallet at /home/h2kim/fabric-samples/e2-list-basic/application-javascript/wal
let/civil
Successfully enrolled admin user and imported it into the wallet
Successfully registered and enrolled user appUser2 and imported it into the wallet

--> Test Generate Public/Private Key Print out fmt.Print
  result: {
  "ID": "pki1",
  "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEA1P7VhVeAEROGZA6DZhth+oZ3dj+KS73
fxIJek2oqZP0w+P5m\nSZp+f400wHryxg8bWibdeL83SEMMb9N6cndmVZXfG7fqAhM0CHFFq7/Za5wqvJJr\nhBFZB5T7CxB3
UFb0nXi12zLMvpYcCM9cBaMnIQzhosA36eJg6vIqA369Bs+7VPje\nRWJcdZv85hhUkBxbpHVTIx/NqJGVur4nuM22ouFXS6j
M53qIDKBf3TiozvRgTq6L\nuvjHI2A47ZOECwg6PMDB32NILOmT3gaU1CnCwE7+KCFk2xEo+WFxjfJ/qjK7i3au\nhwRUwtbD
8B/x+lzGUs6Bemrg4zMcq3gw9lzGEwIDAQABAoIBAEmAh360a2V5L29n\n8sBRwsYJkUAlPTxWEdpmguysPmULWtMSN4o+3lh
2CjZCN3VR0UaTEuqQUiuFt8z/\nVcoamh3+0jdmiEKfDd27+ScD1avlAHqCrMeulOKg7uEqlYIL/+43scorBazfcF05\n8a+g
Veh54YOggascjAWH/CeonOK0O6CADdj7u9rWv00X2I1zy9v54zGKoorJk2vK\nzF/56k39sQNx0l7zDBOaQomFxZwIQqCKc9H
cURqLKJxouQ0cf5j2YLG8Ezy9eu57\nynQ0lyczyQmKHPYom6mnVqzA3bEqs4J8mlDfvmTApumIiGAgCBm68IxdhNKMJX0o\n
a/WiEsECgYEA24iZVVaEDvDOqDq5CqT70b59ijXx76s+0SPMSfSc8msg4o6bt74l\n/suCTbY3mYKYgEig6AQftWMyhI7xV5i
XN/vM46mSMKROIL/FgTDXZ2qmlfIRrybG\npl0/QrbEZbQ2ZGQTrBai1YJCKMPtaOM9DcWjV1LVRPQ7x6nRd+5vgB0CgYEA+G
A1\nWtQi+sRu23Z/XzrEP3bhyhCcY5Z/Edz22E04WOkmV5EZg3L7H66eIyJKsOtOwn5Y\nBtLT9vWO1bLs6RkufOT4phCLZgk
wDgN22yGoQXu/licstFfFX4oUO9WOJyixN5T1\nHIwodJs8udie/80LhkX1Kvo88SUypTlGoYIy5+8CgYEAwnsOq0ULlFjMGs
h66O0j\nkLoaaPhFLjccXRZDgjjLWQ6CgXIojOxMqJ4M/18vfjTteKfes+wDUyNfu4MOVrV9\nMUFGT2EHe0PzWpvJEhv8l0c
p8KgT9oB5dyQgSK+06QxvKxxlOQyMkGzmw0pe5Y55\npduySke9W2qqEsVkJcT6zhkCgYEA6WjgOx8oUCjRVQSi5kc/rbR2s4
GwthX0DEFu\n6A6JaWSRV7FQHKutOzTJ/5J2RpSbS0OYTbFK6MYTGNgws02q1kwHL2hU1+rCcijn\nRT7387HrdZwS8teY8BO
+uAwUrtAa2nTtc4pzakGIa84afaGftut/XogTRTsDwSl2\nKslENZkCgYBg1LUuoIClYocdxCUv8nfgsReHtR63bTmlZyEwuo
iFEhA92JcC/kLz\nImt9UTz2artE5HcnOzEqEBzcCMA6XwUkNmdn5qSRDG4Oa+mmihDyAs8uXTS6hPqN\nSe21ByPA5++XH4a
Lqg01wQrN2s/GynMaUbePl4GifmAAa/cAH+oUmQ==\n-----END RSA PRIVATE KEY-----\n",
  "publicKey": "-----BEGIN RSA PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1P7Vh
VeAEROGZA6DZhth\n+oZ3dj+KS73fxIJek2oqZP0w+P5mSZp+f400wHryxg8bWibdeL83SEMMb9N6cndm\nVZXfG7fqAhM0CH
FFq7/Za5wqvJJrhBFZB5T7CxB3UFb0nXi12zLMvpYcCM9cBaMn\nIQzhosA36eJg6vIqA369Bs+7VPjeRWJcdZv85hhUkBxbp
HVTIx/NqJGVur4nuM22\nouFXS6jM53qIDKBf3TiozvRgTq6LuvjHI2A47ZOECwg6PMDB32NILOmT3gaU1CnC\nwE7+KCFk2x
Eo+WFxjfJ/qjK7i3auhwRUwtbD8B/x+lzGUs6Bemrg4zMcq3gw9lzG\nEwIDAQAB\n-----END RSA PUBLIC KEY-----\n"
}
```

명령이 작동하면서 제대로 javascript-application까지 작동하는 것을 확인할 수 있었음.

```
// Asset describes main asset details that are visible to all organizations
type Asset struct {
    Type  string `json:"objectType"` //Type is used to distinguish the various types of objects in state database
    ID    string `json:"assetID"`
    Color string `json:"color"`
    Size  int    `json:"size"`
    Owner string `json:"owner"`
}

// AssetPrivateDetails describes details that are private to owners
type AssetPrivateDetails struct {
    ID             string `json:"assetID"`
    AppraisedValue int    `json:"appraisedValue"`
}

// TransferAgreement describes the buyer agreement returned by ReadTransferAgreement
type TransferAgreement struct {
    ID      string `json:"assetID"`
    BuyerID string `json:"buyerID"`
}
:%s/transferAgreementObjectType/enterModifiedObjectType/g
```

asset-transfer-private-data에서 사용하는 변수를 진행할 프로젝트에 맞춰서 선언함.

```
// Civil consist of two types of variables. One consists of three states, 'normal', 'suspected', and 'infected', which can be distinguished by a random variabled ID
type Civil struct {
    Type        string `json:"objectType"`
    ID          string `json:"civilID"`
}

// CivilPrivateDetails describes details that personal information to civil
type CivilPrivateDetails struct {
    ID          string  `json:"civilID"`
    Name        string  `json:"civilName"`
    PhoneNumber string  `json:"civilPhoneNumber"`
    Address     string  `json:"civilAddress"`
}

// Shop is consist of four variables. first identifier of store. second name of store. third telephone-number. fourth address of shop.
type Shop struct {
    ID          string `json:"shopID"`
    Name        string `json:"shopName"`
    Telephone   string `json:"shopTelephone"`
    Address     string `json:"shopAddress"`
}
```

위와 같이 선언을 하고 체인코드 작성시 변경사항이 있으면 상황에 맞추어 변경하기로 함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git commit -m "first commit"
[master (최상위-커밋) 76ed209] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git branch -M main
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git remote add origin https://github.com/
ik2ki/e2-list-private-data.git
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git push -u origin main
Username for 'https://github.com': ik2ki
Password for 'https://ik2ki@github.com':
오브젝트 개수 세는 중: 3, 완료.
오브젝트 쓰는 중: 100% (3/3), 229 bytes | 229.00 KiB/s, 완료.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ik2ki/e2-list-private-data.git
 * [new branch]      main -> main
'main' 브랜치가 리모트의 'main' 브랜치를 ('origin'에서) 따라가도록 설정되었습니다.
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ ls
README.md  application-javascript  chaincode-go
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git add application-javascript
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git add chaincode-go
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git commit -m "application-javascript and
 chaincode"
[main 9b9c318] application-javascript and chaincode
 951 files changed, 368288 insertions(+)
 create mode 100644 application-javascript/.eslintignore
 create mode 100644 application-javascript/.eslintrc.js
 create mode 100644 application-javascript/.gitignore
 create mode 100644 application-javascript/app.js
 create mode 100644 application-javascript/app.js.bak
 create mode 100644 application-javascript/package.json
 create mode 100644 application-javascript/wallet/org1/admin.id
 create mode 100644 application-javascript/wallet/org1/appUser1.id
 create mode 100644 application-javascript/wallet/org2/admin.id
 create mode 100644 application-javascript/wallet/org2/appUser2.id
 create mode 100644 chaincode-go/META-INF/statedb/couchdb/collections/assetCollection/indexes/ind
exOwner.json
 create mode 100644 chaincode-go/README.md
 create mode 100644 chaincode-go/chaincode/asset_queries.go
 create mode 100644 chaincode-go/chaincode/asset_queries_test.go
 create mode 100644 chaincode-go/chaincode/asset_transfer_test.go
 create mode 100644 chaincode-go/chaincode/e2_list_enter.go
 create mode 100644 chaincode-go/chaincode/mocks/chaincodestub.go
 create mode 100644 chaincode-go/chaincode/mocks/clientIdentity.go
 create mode 100644 chaincode-go/chaincode/mocks/statequeryiterator.go
 create mode 100644 chaincode-go/chaincode/mocks/transaction.go
 create mode 100644 chaincode-go/collections_config.json
 create mode 100644 chaincode-go/go.mod
 create mode 100644 chaincode-go/go.sum
```

체인코드 개발할 때 장애가 발생하면 복구를 위해 git을 통해서 체인코드를 업데이트 하기로 함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data$ git push -u origin main
Username for 'https://github.com': ik2ki
Password for 'https://ik2ki@github.com':
오브젝트 개수 세는 중: 1115, 완료.
오브젝트 압축하는 중: 100% (1017/1017), 완료.
오브젝트 쓰는 중: 100% (1115/1115), 2.30 MiB | 2.23 MiB/s, 완료.
Total 1115 (delta 215), reused 0 (delta 0)
remote: Resolving deltas: 100% (215/215), done.
To https://github.com/ik2ki/e2-list-private-data.git
   76ed209..9b9c318  main -> main
'main' 브랜치가 리모트의 'main' 브랜치를 ('origin'에서) 따라가도록 설정되었습니다.
```

git push를 통해서 github에 소스코드가 올라간 것을 확인할 수 있음.

비교할 수 있는 대상, 비교 항목을 찾기 위해 다른 논문을 찾아보고 정리 이후 프로젝트 진행에 따라서 적용할 수 있는 대상과 항목을 접합하여 논문을 작성하는데 유의미한 결과를 낼 수 있도록 논문 검색을 통해 다른 연구에서 무엇을 비교하고 어떤 결과를 도출했는지 찾아보았습니다.

참고자료[1]의 하이퍼레저와 MySQL을 비교한 논문에서는 노드의 숫자에 따른 처리율과 지연율을 결과를 냈습니다.

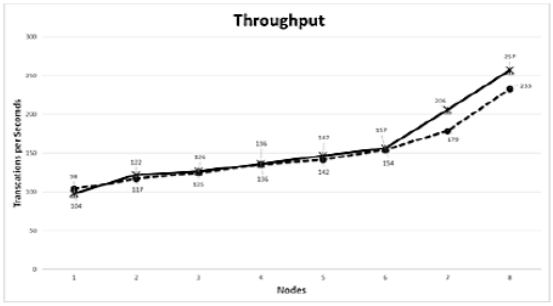

Figure 3. Throughput

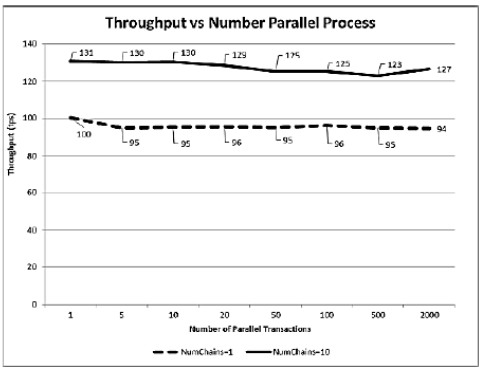

Figure 5. Throughput vs number parallel process

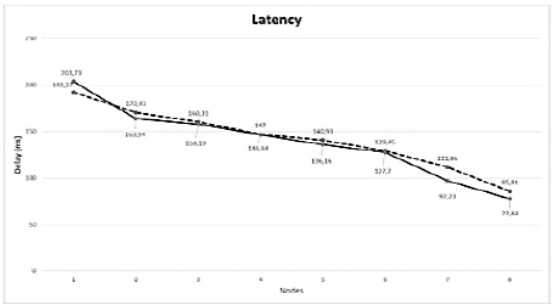

Figure 4. Latency

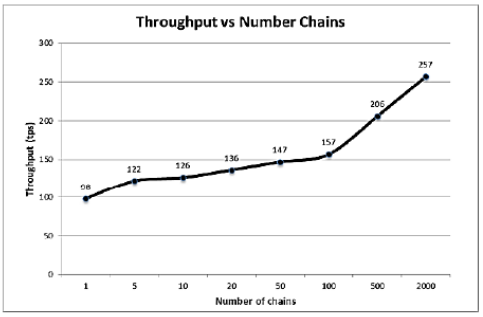

Figure 6. Throughput vs number chains

병렬처리 숫자에 따른 처리량를 비교하는 결과를 낸 것도 볼 수 있었습니다. 체인의 개수, 블록당 트랜잭션 수, 블록크기, 트랜잭션이 블록에서 일어나는 숫자에 따른 처리율을 가지고 그래프를 작성했습니다. 데이터 크기에 따라서 읽기/쓰기의 DLT과 SQL의 시간차이, 처리량을 비교한 것을 볼 수 있었습니다.

참고자료[2] 의료자료를 공유하기 위해 하이브리드 블록체인을 적용하는 제안을 하는 논문입니다. 공개블록체인, 합의 블록체인, 비공개블록체인을 비교한 후 의료정보 시스템에는 하나의 방법으로 해결하는 방법보다는 특장점을 이용해 적용할 수 있는 분야가 다르다는 것을 서술했습니다.

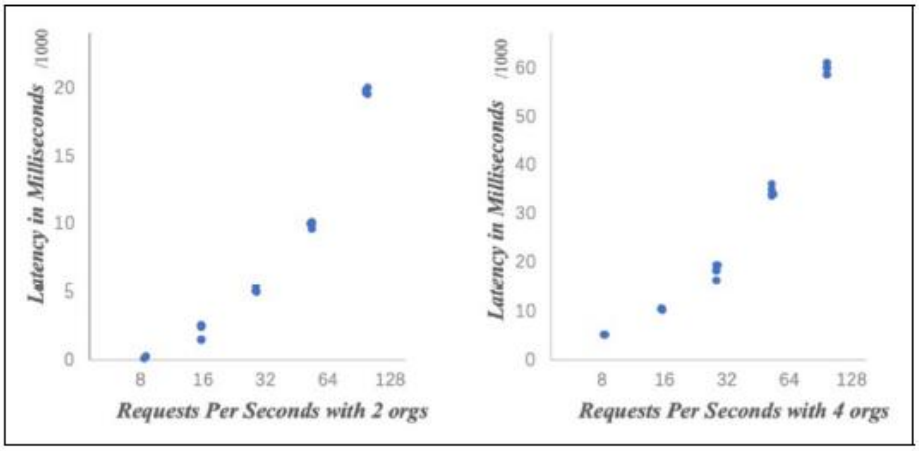

Figure 10. HB-EMRS response latency comparison deployed with two and four organizations.

성능평가는 2개의 조직 4개의 조직으로 지연율을 측정했습니다.

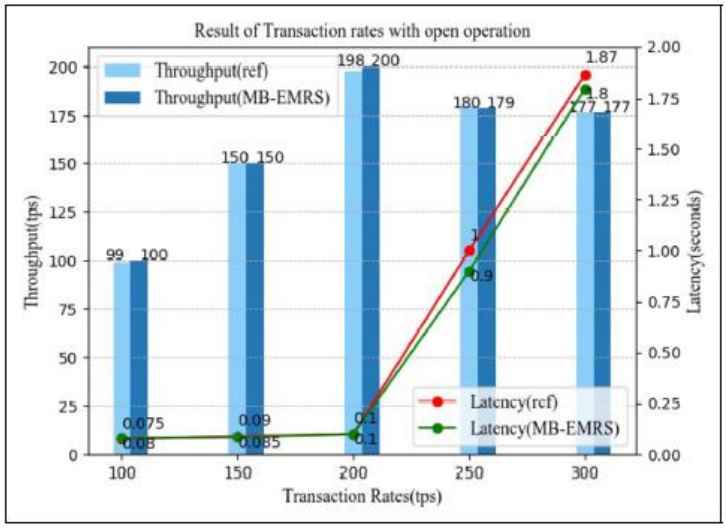

Figure 11. Result of transaction rates with open operation.

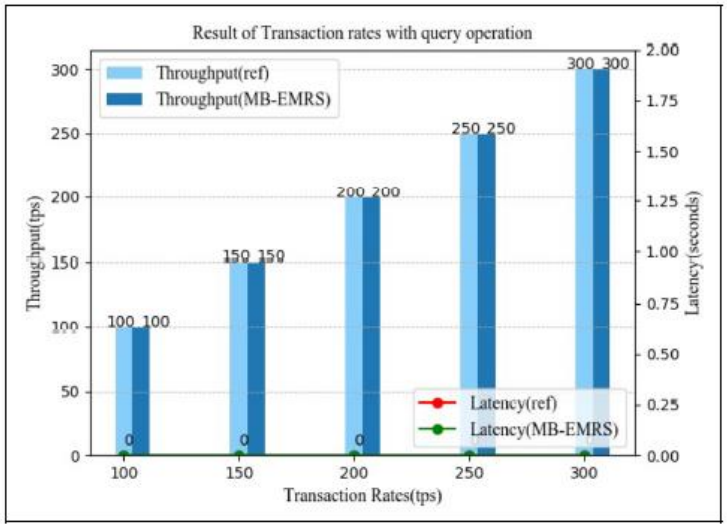

Figure 12. Result of transaction rates with query operation.

논문에서 제안한 모델의 성능측정을 위한 비교는 트랜잭션속도는 100, 150, 200, 250, 300 tps로 하고 트랜잭션의 개수는 1000개로 고정했습니다. 트랜잭션 숫자의 영향을 보기 위해서 트랜잭션 속도는 200tps로 고정하고 트랜잭션 숫자는 1000, 10,000, 100,000로 정했습니다. 실험에 따른 결과는 처리량과 지연율을 가지고 결과를 내는 것을 알 수 있었습니다.

참고자료[3]의 goleveldb의 성능에 관한 논문에서는 다음과 같은 비교했습니다.

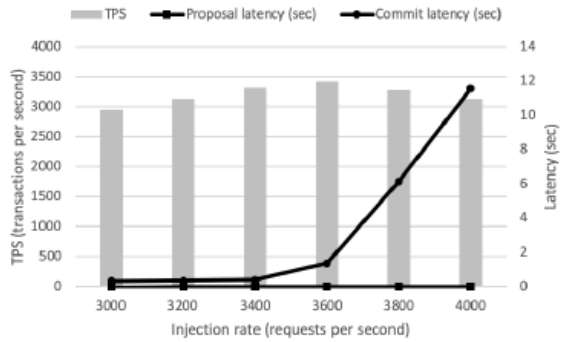

Fig. 2. TPS (Transactions Per Second) and latency of a bank chaincode with 256-byte values.

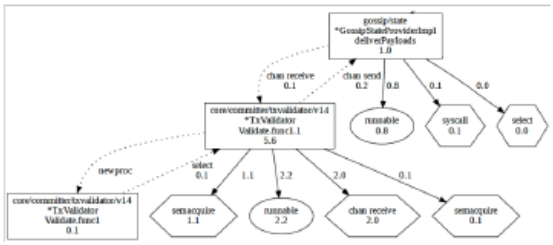

Fig. 3. Thread dependency graph of a bank chaincode with 256-byte values.

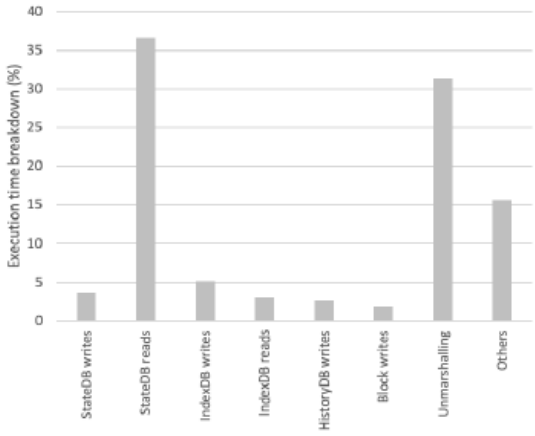

Fig. 4. Execution time breakdown of a bank chaincode with 256-byte values.

요청에 따라서 처리량, 제안지연시간(proposal latency), 작업완료 지연시간(commit latency)를 구했습니다.

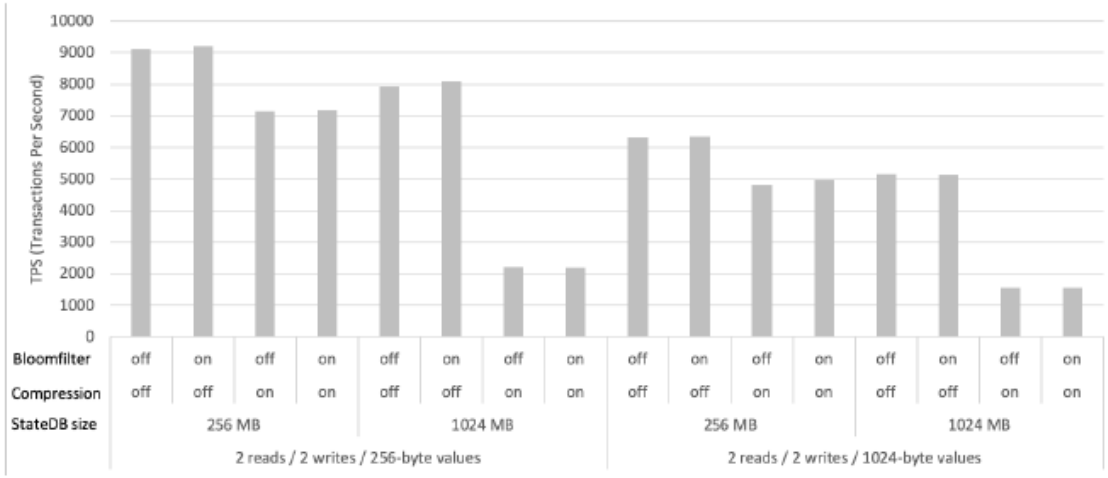

Fig. 6. Simulated transactions per second (TPS) of Hyperledger Fabric via HLF-GLDB with two reads and two writes in a transaction.

블룸필터(Bloomfilter), 압축(Compression), 스테이트DB크기(StateDB size)을 다르게 설정하여 처리량을 비교했습니다.

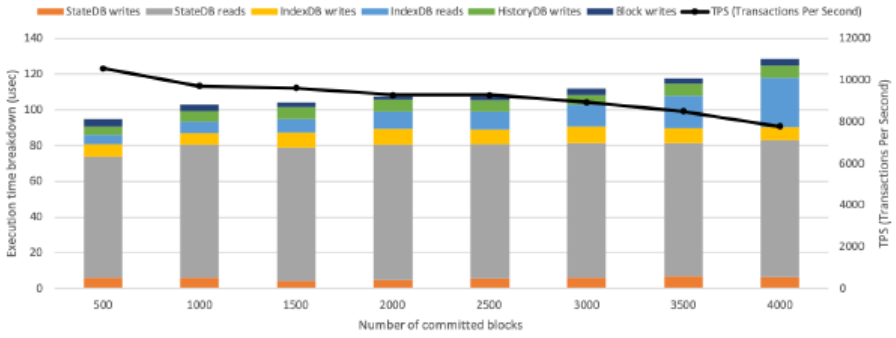

Fig. 7. Execution time breakdown and simulated transactions per second (TPS) of Hyperledger Fabric via HLF-GLDB with two reads and two writes in a transaction. Value size of each read and write is 256 bytes. StateDB size is 256 MB. Compression is disabled and bloom filter is enabled.

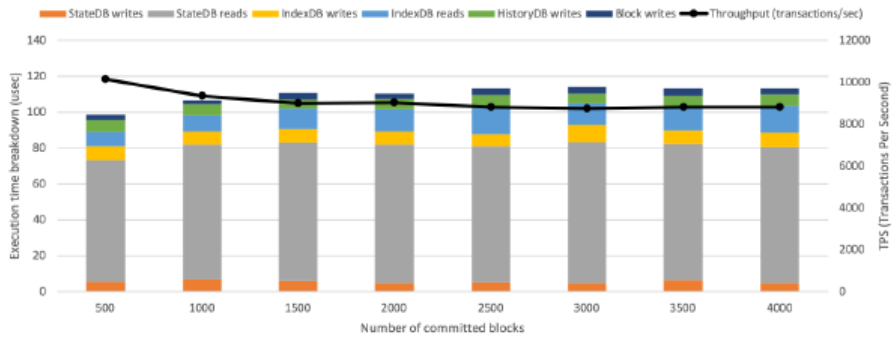

Fig. 8. Execution time breakdown and simulated transactions per second (TPS) of Hyperledger Fabric via HLF-GLDB with two reads and two writes in a transaction. Value size of each read and write is 256 bytes. StateDB size is 256 MB. Compression is disabled and bloom-filter is disabled.

처리완료블록(committed block)의 개수를 가지고 숫자를 늘려가면서 처리량을 구하고 처리완료시간을 구성하고 있는 각 단계의 걸리는 시간을 분해하여 그래프로 작성했습니다.

참고자료[4]에서는 3가지의 케이스를 정해놓고 처리량과 지연량 확장성을 구하

는 실험을 구성했습니다.

TABLE I. PARAMETERS FOR PERFORMANCE EVALUATION

Key parameters	Transaction per second (tps)	Number of transactions	Simul-taneous transactions
Case I: Impact of test environment and transaction rates	100, 150, 200, 250, 300tps	1,000	N/A
Case II: Impact of number of transactions	200tps	1,000 10,000 100,000	N/A
Case III: Impact of simultaneous transactions	Equivalent to number of simultaneous transactions	N/A	100 200 300

트랜잭션 지연시간이 영향을 실험에 영향을 주는 첫 번재 유형, 트랜잭션 숫자가 영향을 주는 두 번재 실험, 마지막으로 동시거래가 영향을 미치는 마지막 케이스가 있습니다. 각 결과는 다음의 삽입한 표와 같습니다.

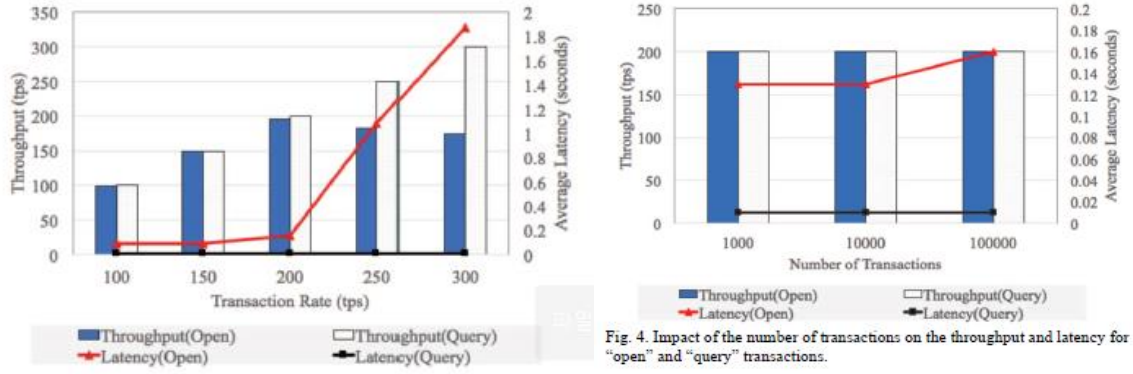

Fig. 4. Impact of the number of transactions on the throughput and latency for "open" and "query" transactions.

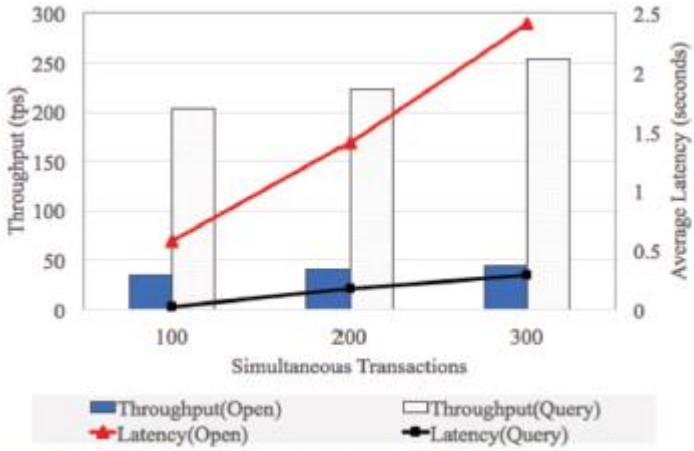

Fig. 5. Impact of the number of participants on throughput and latency.

3. 차주 계획

- e2-list-private-data 프로젝트 체인코드 함수 작성 테스트코드 작성

- 성능측정 논문 검색 정리 적용 고민

4. 참고 자료

[1] Onno W. Purbo, Sriyanto Sriyanto, Suhendro Suhendro, Rz Abd. Aziz, and Riko Herwanto "Benchmark and comparison between Hyperledger and MySQL" TELKOMNIKA Telecommunication, Computing, Electronics and Control Vol. 18, No. 2, April 2020, pp. 705~715

[2] Yu Cao, Yi Sun, Jiangsong MIN "Hybrid blockchain–based privacy-preserving electronic medical records sharing scheme across medical information control system" Measurement and Control Volume: 53 issue: 7-8, pp. 1286-1299

[3] Takuya Nakaike, Qi Zhang, Yohei Ueda, Tatsushi Inagaki, and Moriyoshi Ohara "Hyperledger Fabric Performance Characterization and Optimization Using GoLevelDB Benchmark" 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp 1-9

[4] Murat Kuzlu, Manisa Pipattanasomporn, Levent Gurses and Saifur Rahman "Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability" 2019 IEEE International Conference on Blockchain (Blockchain), pp 536-540