<블록체인 기반의 네트워크 접근제어 시스템>

영남대학교 기계IT대학 컴퓨터공학과

21511748 박준영

21511712 강한샘

21511759 신윤상

21511781 이찬영

제출일 : 2020/11/25

<제목 차례>

<표 차례>

<그림 차례>

서론

본 보고서는 기존의 중앙 집중방식의 네트워크 접근제어 시스템에서 벗어나 무결성 및 보안성을 증대시킬 수 있는 네트워크 접근제어 시스템을 구현하기 위해 뛰어난 보안성과 분산처리로 최근 주목받고 있는 블록체인과 스마트 컨트랙트 기술을 이용한 네트워크 접근제어 시스템을 설명한다. 블록체인과 스마트 컨트랙트를 이용함으로써 기존에 에이전트를 통해 처리하던 작업을 블록체인을 통해 동작하게 함으로써 사용자에게 에이전트의 설치를 요구하지 않으며 네트워크 구축을 위한 필요 장비 감소 및 시스템 자체의 임의 변조 등 무결성을 증대시킬 수 있게 되었다.

보고서에서 설명하는 시스템은 기존의 네트워크 접근시스템이 제공하는 기능에 더불어서 최근 IoT 장비가 증가하는 추세에 맞춰 IoT 장비에 대한 접근제어 또한 지원할 수 있도록 설계되었다. IoT 장비는 입출력 형식이 제조사마다 다르고 IoT 장비의 특성상 시스템을 이용하는 것에 제한상황이 발생할 수 있으니 시스템을 자유자재로 이용할 수 있는 Raspberry PI를 통해 일반 전자 기기부터 IoT 장비까지 제어가 가능한 게이트웨이를 개발하여 시스템의 확장성 또한 크게 증가하였다. 또한 네트워크에 참여하기 위해 조직의 신원 확인과 PC, 스마트폰, IoT 기기들을 포함한 모든 기기가 블록체인 네트워크에 접근하기 위한 웹 페이지, 웹 클라이언트를 구축함으로 시스템에 접근성을 향상하고 사용성을 증대하였다.

이러한 내용을 종합하여 앞서 설명한 내용에 대한 시스템 구조와 하드웨어 및 소프트웨어 구현 방법, 알고리즘, 테스트 방법, 결과 및 결론 등을 본 보고서에 서술하였다.

1. 시스템 개요

1.1. 설계 개요

블록체인 및 스마트 컨트랙트를 이용하여 비 중앙방식의 네트워크 접근제어 시스템을 설계하였다. 해당 시스템은 기존의 블록체인 네트워크를 사용하여 기존의 네트워크 접근제어 방식에서 활용된 에이전트 및 보안정책을 위한 네트워크 장비를 사용하지 않게 설계되었다. 또한 일반 PC부터 스마트폰, IoT 이르는 기기들을 접근 제어할 수 있는 확장성을 가지기 위해 Raspberry PI를 통한 게이트웨이 및 웹 페이지, 웹 클라이언트들을 설계하였다. 위의 내용을 통해 기존 네트워크 접근시스템에서 IPv6 및 IoT 장비에 대한 네트워크 접근제어를 위한 문제점인 보안성 유지를 위한 중앙 서버의 부하와 별도의 장비 추가를 위한 비용증가, 보안성 약화 등의 문제점을 해소할 수 있으며 블록체인 기반 시스템의 장점인 부하 분산, 데이터에 대한 무결성, 가용성의 증대를 가져올 수 있다.

1.2. 제한조건

해당 시스템은 기기들의 모든 자원에 대해 접근제어가 가능하도록 설계되어 있으나 자원의 종류는 무한하므로 테스트 및 구현을 위해 블록체인 네트워크를 제외한 웹 클라이언트 및 게이트웨이는 인터넷 자원만을 제어하도록 설계되었다.

1.3. 역할 분담

이름: 박준영	학번: 21511748
● 게이트웨이 설계 ● 게이트웨이 구현	

표 1 역할분담-박준영

이름: 이찬영	학번: 21511781
● 블록체인 네트워크 설계 ● 블록체인 네트워크 구현 ● 프런트 서버 설계 ● 프런트 서버 구현 ● 백앤드 서버 설계 ● 백앤드 서버 구현	

표 2 역할분담 이찬영

이름: 강한샘	학번: 21511712
● 스마트컨트랙트 설계 ● 스마트컨트랙트 구현	

표 3 역할분담 강한샘

이름: 신윤상	학번: 21511759
● 게이트웨이 설계 ● 게이트웨이 구현	

표 4 역할분담 신윤상

2. 시스템 기능

본 시스템에서 제공하는 기능들은 크게 접근통제, 보안정책 관리, 사용자 관리, 이상 행동 감지, 통신장비 및 IoT 장비 연결 가능, 유저 인터페이스 등으로 구성되어 있다.

2.1. 보안정책 관리

시스템에서 관리할 자원에 대해 보안정책들을 권한이 있는 관리자가 웹 클라이언트를 통해 블록체인 네트워크에 등록, 갱신, 삭제하는 기능을 제공한다.

2.2. 사용자 관리

시스템에 접근할 수 있는 사용자들의 관리에 대한 방법을 제공한다. 관리자가 웹 클라이언트를 통해 블록체인 네트워크에 등록, 갱신, 탐색, 삭제하는 기능들을 제공한다.

2.3. 접근통제 기능

사용자가 블록체인 네트워크를 통해 특정 자원에 접근하려 할 때 게이트웨이는 블록체인 네트워크와의 질의를 통해 현재 정책을 확인하여 사용자의 접근을 허용 통제 한다.

2.4. 이상 행동 감지

시스템을 이용하는 있는 사용자가 보안정책에 위배되는 특정 행동을 감지하여 해당 사용자가 보안정책에 위배되는 행동을 하는지를 감시하여 탐지 시 해당 사용자에게 패널티를 부과할 수 있는 기능을 제공한다.

2.5. 통신장비 및 IoT 장비 연결 가능

PC, 스마트 및 IoT 장비들의 연결을 제공한다. IoT 장비들의 특징인 다양한 제조사와 기능의 한계를 고려하여 Raspberry Pi를 통해 장비들을 관리하고 블록체인 네트워크 시스템을 이용할 수 있도록 한다.

2.6. 유저인터페이스

본 시스템에서 이용되는 기능들을 사용자와 관리자가 권한에 따라 특정 기능들을 쉽게 수행할 수 있도록 User Interface를 FrontedServer, BackendServer를 통해 제공하여 사용자들의 편의성을 증대시킬 수 있도록 한다.

3. 블록별 기능 구현

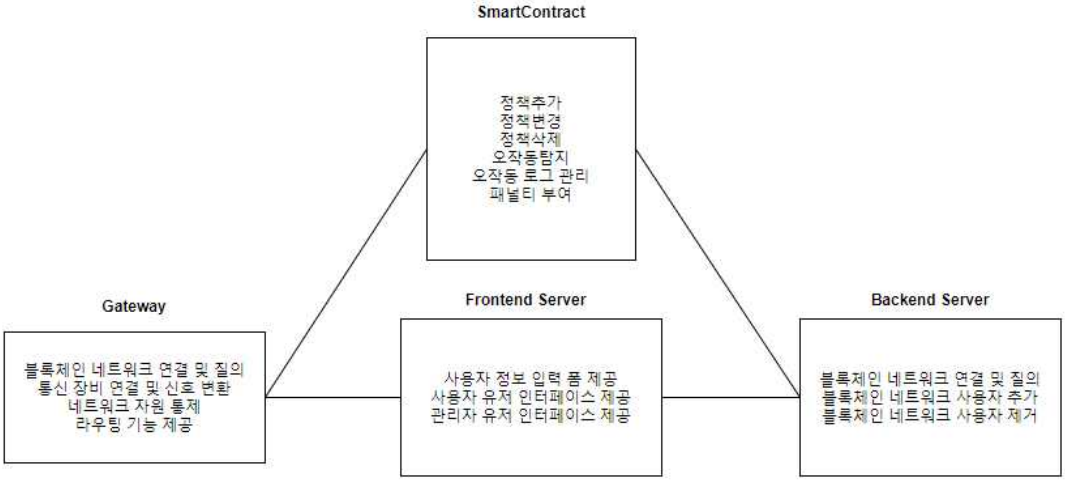

그림 1 블록다이어그램

위 그림은 보고서에서 설명하는 시스템의 블록 다이어그램이다. 기능별로 블록은 총 3가지가 있으며 각각 게이트웨이, 스마트 컨트랙트, Frontend Server, Backend Server이다. Backend Server는 블록체인 네트워크의 클라이언트 역할을 담당한다. Rest API를 통해 게이트웨이, Fabric Network, Frontend Server와 통신을 한다. Frontend Server는 NAC 서비스를 사용하는 조직이 자신이 공유할 네트워크에 대한 사용자 정보나 접근 제어할 체인코드 정보를 수정하는 페이지를 제공한다.

게이트웨이는 라즈베리파이로 만들어지며 Backend 서버와 연결되어 권한에 대한 질의를 하게 되며 이외에도 실질적인 네트워크 자원의 통제, 통신 장비들(PC, 스마트폰, IoT) 연결과 신호변환, 정상 작동 시의 라우팅 기능을 제공한다. 스마트컨트랙트는 보안 정책의 추가, 변경, 삭제의 기능을 하며 오작동 탐지, 오작동 로그 관리, 패널티 부여 기능을 한다.

4. 시스템 구조

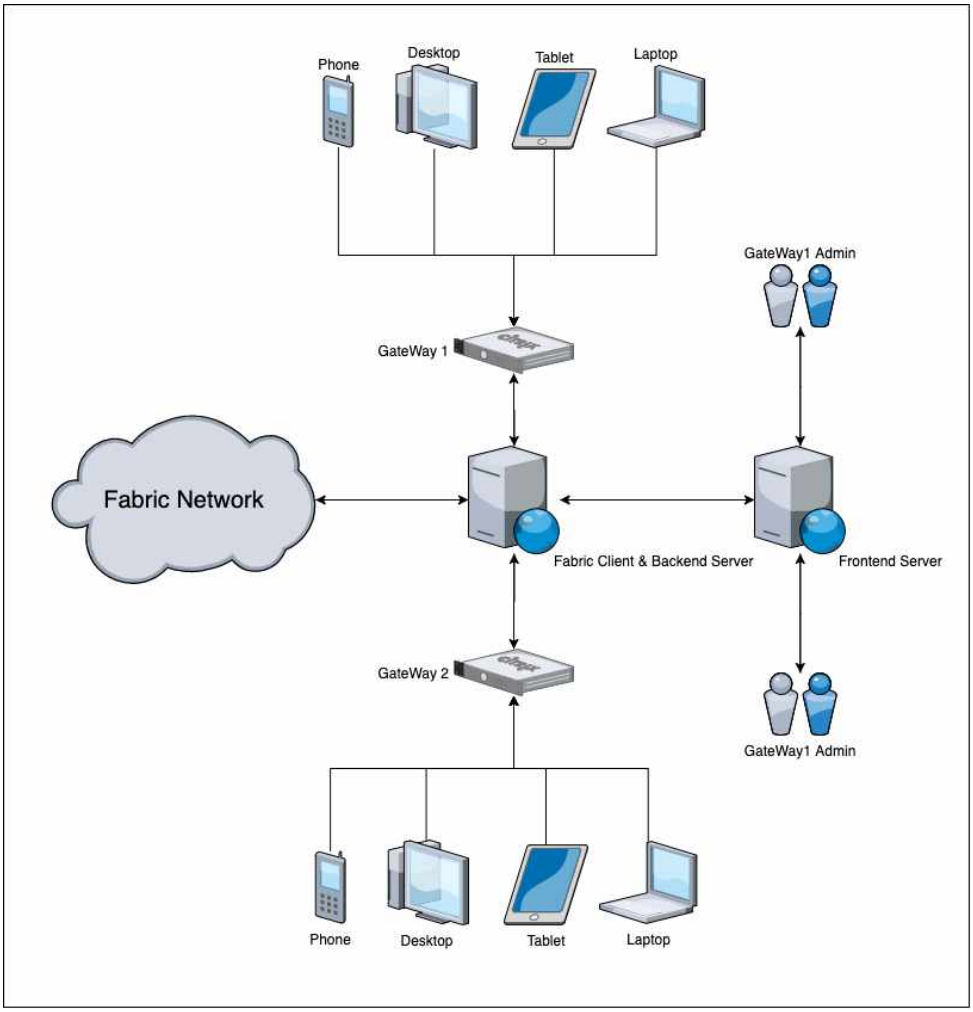

그림 2 시스템 구성도

전체 시스템 구조는 크게 블록체인(Fabric) 네트워크, Fabric Client & Backend Server, Frontend Server, 게이트웨이로 구성되어 있다. Fabric Network는 접근제어 시스템이 체인코드로 구현되어있다. Fabric Network는 Fabric Client인 Backend Server와 모든 통신이 이루어진다. Backend Server는 Restful API로 Front Server와 게이트웨이와 통신을 하게 된다. Frontend Server는 채널 생성, 체인코드 업로드, 공유할 네트워크 추가, 네트워크 사용자 추가 등 공유할 네트워크를 관리할 수 있는 페이지를 구현하였다. 게이트웨이는 실질적인 자원의 통제를 하는 역할을 하며 Fabric Client로 Fabric Network와 통신을 하게 된다.

4.1. 블록체인 네트워크

BlockChain의 스마트컨트렉트(체인 코드)를 이용하여 클라이언트 단에서는 추가적인 Agent 설치 없이 네트워크에서 접근제어 시스템을 구현하였다. 이를 구현하기 위해 HyperIdeger Fabric이라는 Private BlockChain을 사용하였다. 이더리움과 같은 Public BlockChain이라면 허가되지 않은 사용자가 체인코드를 호출하여 무단으로 접근 권한 값을 변경하거나 공유되는 네트워크의 정보의 내용을 가로챌 수 있는 위험이 있다. 하지만 Private Block Chain을 이용하여 허가된 조직의 관리자만 체인코드의 호출을 할 수 있게 된다. 네트워크를 공유받는 사용자나 그렇지 않은 외부인 모두 체인코드 호출을 할 수 없게 됨으로써 공유되는 네트워크의 접근 권한 상태 값은 외부인의 개입 없이 안전하게 저장될 수 있다.

4.1.1. HyperLedger Fabric이란 ?

Hyperledger Fabric은 여러 조직과 신뢰를 바탕으로 이루어진 분산 저장시스템이다. 신뢰하는 조직끼리 서로가 합의된 체인코드로 불변하는 데이터를 저장한다. 일반 관계형 데이터베이스의 분산 저장시스템과 차이점은 하이퍼레저 페브릭은 데이터를 저장하는 방식이 블록체인이 된다. 블록체인은 모든 트랜잭션을 해시 하여 레코드와 연결이 된다. 이전 레코드를 변경하면 체인은 손상이 된다. 그러나 일반 관계형 데이터베이스는 그렇지 못하다. 하이퍼레저 페브릭은 블록체인을 이용한 분산 저장시스템이라고 보면 된다.

Hyperledger 구성요소 중 네트워크에 참여하는 노드를 간단히 설명하겠다. 노드는 각각의 역할이 있으며 피어, 오더러로 나눌 수 있다.

피어는 체인코드를 유지 관리하고 트랜잭션을 승인하며 트랜잭션 및 블록의 유효성을 검사한다. 쉽게 말해 체인코드의 복사본을 가지고 있으며 각각의 모든 피어가 장부를 가지고 있어 블록이 변조됐는지를 확인하는 역할을 한다. 피어 안에도 종류가 있으며 리더 피어, 앵커 피어, 보증 피어, 검증 피어가 있다.

리더 피어란 오더러에게 트랜잭션의 순서가 정리된 블록을 받아서 조직 내부의 검증 피어들에게 배포한다.

앵커 피어란 채널 외부에서 조직 외부의 네트워크에 있는 다른 노드와 통신을 하는 노드이다. 예를 들어 채널 1에 A와 B, C 조직이 있다. 이 채널에서 A 조직 중 노드 하나를 앵커 피어로 설정하게 되면 A 조직의 앵커 피어 하나로 B, C 조직의 노드들을 알 수 있게 된다.

다음으론 보증 피어이다. 보증 피어는 클라이언트에서 보낸 트랜잭션을 블록으로 만들기 위한 보증 응답(endorsed response)을 생성한다. 보증 응답은 오더러 노드로 전송이 되며 유효한 피어들에게 보내집니다. 또한 체인코드를 호스팅하고 체인코드를 실행하여 트랜잭션을 확인하고 결과를 클라이언트로 반환한다.

마지막으로 검증 피어입니다. 검증 피어는 오더러 노드가 블록에서 트랜잭션 순서를 정리하면 각 조직에서 선출된 리더 피어에게 트랜잭션을 보내고 이후 리더 피어가 트랜잭션을 검증하기 위해 모든 검증 피어에게 분배 후 트랜잭션에 이상이 없다면 원장에 새로운 결과를 등록한다. 이 검증 피어는 모든 피어가 될 수 있으며 한 피어는 여러 역할을 수행할 수 있다.

채널은 CC라는 Channel Consortium으로 구성된 블록체인 네트워크라고 보면 된다. 채널

은 여러 조직이 하나의 네트워크를 구성하는 것이며 채널은 큰 블록체인 네트워크 안의 서브체인(서브 네트워크)가 되는 것이다. 각 채널은 독립적이며 각각의 체인코드와 장부를 가지고 있다. 클라이언트에서 채널 안의 조직에서 발급한 인증서가 없다면 체인코드를 호출할 수 없다.

위에서 설명한 내용이 버전이 바뀌면서 다른 내용이 있을 수 있기 때문에 정확한 내용은 버전에 맞게 찾아보는 것을 추천한다. 이것 외에도 더 많은 구성요소들도 있고 이해를 위해선 하이퍼레저에 대한 구조를 알아야 할 필요성이 있다.

4.1.2. 블록체인 네트워크 구조

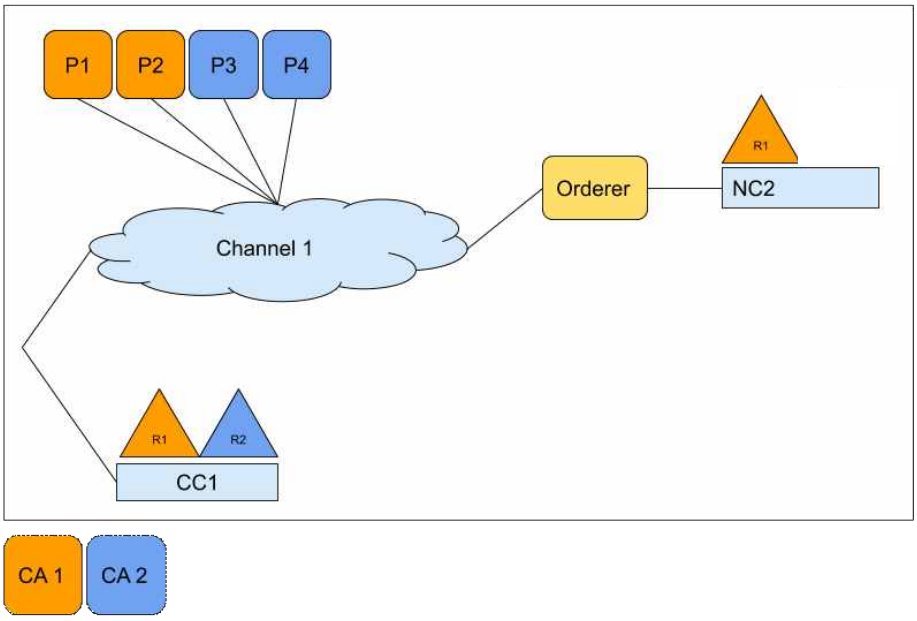

그림 3 블록체인 네트워크 모델

현재 프로젝트에서 구현된 블록체인 네트워크 모델이다. 그림에서 주황색은 NAC Service를 제공하는 우리 프로젝트이며 파란색은 NAC Service를 사용하는 네트워크 공유자이다. 이제부턴 색이 아닌 가상의 조직에 빗대어 설명하겠다. 주황색은 NAC 구축 회사이며, 파란색은 NAC 서비스를 이용하는 영남대이다.

다음으로 R1, R2 세모는 각각 조직(NAC구축 회사와 영남대)을 뜻하며, P1~P4 주황색, 파랑색 네모는 피어를 뜻한다. CA1, CA2는 CA 인증기관이다. 채널에서 CC는 Channel Consortium로 채널을 구성하는 조직 컨소시엄이라고 보면 된다. Channel 1에서는 NAC 구축 조직과 영남대 조직으로 이루어져 있다. Orderer 또한 두 조직이 모두 참여한다. NC는 Network configuration으로 전체 네트워크의 관리자 권한을 가진 조직들이다. NAC 구축 회사 조직에서만 전체 네트워크를 관리할 수 있도록 하였다.

Channel 1은 NAC 구축 회사 조직과 영남대 조직으로 구성되어 있으며 각각의 조직에서 2개씩 피어가 참여하고 있다. 블록체인 네트워크에 참여 된 노드의 수가 적기 때문에 4개의 노드만 조작하면 블록체인 네트워크를 쉽게 조작할 수 있다고 생각하겠지만 프라이빗 블

록체인 특성상 각 조직끼리 이미 신뢰를 형성하고 있다. NAC 서비스를 구축하는 조직은 영남대조직에게 일정한 돈을 받을 것이고 해당 고객을 유치하기 위해 NAC 서비스 구축하는 조직은 해당 피어들의 장부를 조작하진 않을 것이다. 또한 영남대 조직에게 악의적인 행동을 하기 위해 관리자 피어 장부를 조직할 순 있겠지만 NAC 구축 조직의 피어들로 인해 장부 조작은 불가능하게 된다. 이러한 프라이빗 블록체인의 특성으로 인해 네트워크에 많은 노드를 참여시키지 않더라도 정확하고 빠른 시간에 트랜잭션들을 처리할 수 있게 된다.

위에서 본 형태는 하나의 NAC 서비스 사용자 조직으로만 서비스를 구현했을 때의 모습이고 다음은 조직이 하나 더 추가된 모습을 보겠다.

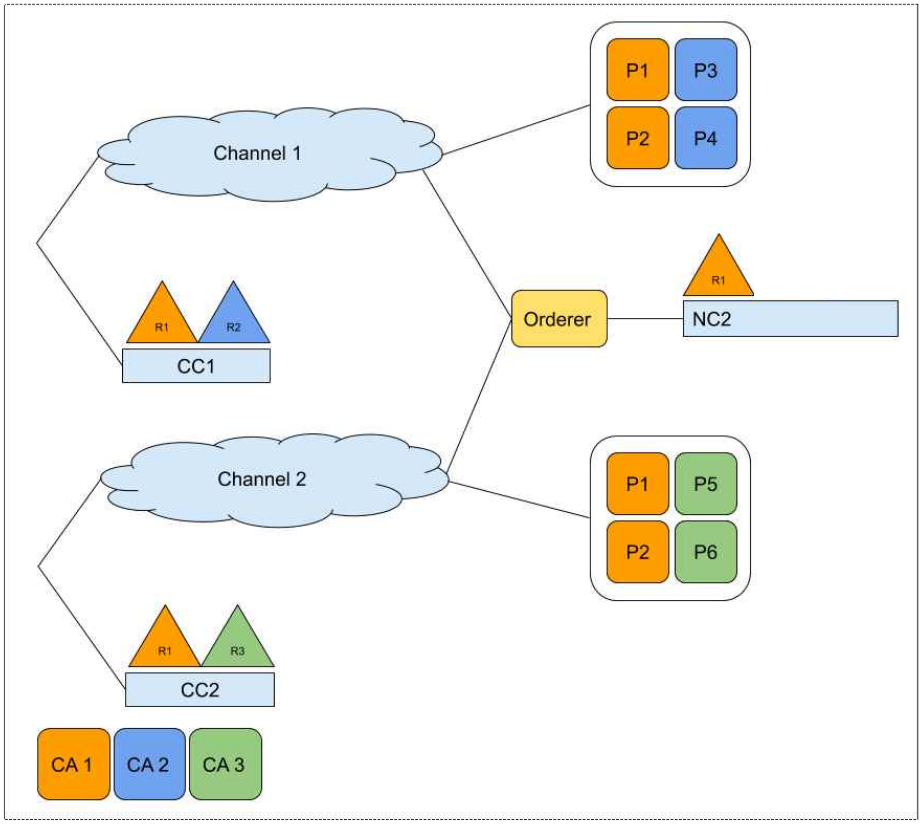

그림 4 채널 2가 연결된 모습

추가된 조직은 기본 형태와 똑같고 오더러에 채널2가 연결된 모습이다. 초록색이 경북대 조직이라고 하자. 채널 2는 NAC 구축 조직과 경북대 조직이 서브 네트워크를 형성한 모습이며 채널 1에서 저장된 장부와 체인코드와는 완전히 독립적으로 존재한다. 그렇기 때문에 해당 조직에 맞는 접근제어 체인코드를 동작시킬 수 있다. 예를 들어 채널1에서는 IOT 기기들에 접근하는 접근제어 체인코드를 올릴 수 있으며 채널 2에서는 PC들의 네트워크를 공유하는 접근제어 체인코드를 올릴 수 있다. 또한 채널1의 관리자는 자신의 채널에서만 관리자이므로 채널2의 체인코드를 실행시킬 순 없다.

4.2. 스마트 컨트랙트

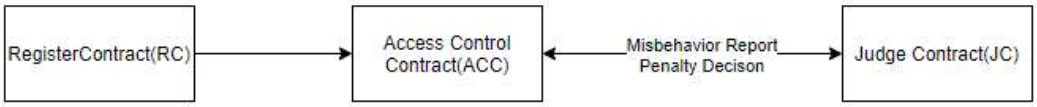

그림 5 스마트컨트랙트 시스템

접근 제어 시스템에 필요한 스마트컨트렉트는 RC, ACC, JC 3개의 체인코드로 구성되어 있다. RC는 네트워크 자원을 공유할 조직과 자원의 접근하려는 사용자들과 자원을 관리한다. ACC는 해당 사용자가 네트워크의 자원에 대한 권한 정책을 관리하며 네트워크 자원에 접근 요청, 오작동에 대한 차단 로그 관리 기능을 한다. JC는 오작동에 대한 패널티를 부여한다. 체인코드는 Java로 작성하였다. 아래에선 각 체인코드를 자세히 설명한다. 아래의 설명에서 블록체인 네트워크와 네트워크는 다른 의미를 가지고 있으며, 블록체인 네트워크는 말 그대로 Fabric Network를 뜻하며 네트워크는 NAC 구축 서비스를 사용하는 사람들이 공유할 네트워크를 뜻한다 (예를 들면 위에서 설명한 영남대를 뜻함).

4.2.1. RegisterContract(RC)

RC는 공유할 네트워크와 네트워크를 사용하는 사용자들의(예를 들어 공유하는 게이트웨이와 그에 물려있는 단말 노드 기기들) 정보를 관리하는 체인코드이다. 해당 네트워크에 연결된 노드들을 추가, 수정, 삭제할 수 있다. 예를 들어 영남대 조직 관리자는 IT관 1층을 관리하는 게이트웨이 정보를 LookUpTable에 저장할 수 있으며 해당 게이트 웨이에 물려있는 PC1, PC2 ... PC30 이라는 사용자(단말 노드) 정보를 저장할 수 있다. 노드들을 추가 수정, 삭제함으로써 해당 네트워크를 공유하는 게이트웨이에서 LookUpTable에 등록되지 않은 단말기기 들이 있다면 확인 후 네트워크에 접근할 수 없도록 차단할 수 있게 한다. 또한 ACC 체인코드의 정보를 담고 있어 ACC 체인코드 업데이트를 빠르게 반영할 수 있다. 룩업테이블에 구조는 아래와 같다.

Name	Type	Description
ID	String	랜덤으로 생성한 UUID
MethodName	String	네트워크를 설명할 이름
Subject	Object	게이트 웨이의 이름과 맥주소를 가지는 객체
Objects	Array	각각의 오브젝트의 이름과 맥주소를 가지는 객체의 배열
scName	String	Acc 체인코드의 이름
abi	String	Acc 체인코드에서 실행할 접근제어 함수 이름

표 5 RC 체인코드의 LookUpTable 구조

4.2.2. Access Control Contract(ACC)

ACC의 PolicyTable은 하나의 공유 네트워크(Rc 체인코드의 LookUpTable의 한 row)와 하나의 노드 단말기기로 맵핑된다. 예를 들어 IT관 1층의 게이트웨이와 PC1이 맵핑되는 구조이다. ACC는 사전 정의된 접근 권한과 단말 노드 기기에서의 비정상적 접근 동작을 확인하여 동적으로 접근 유효성 검증을 할 수 있다. 비정상적 접근 동작을 하게 되면 로그가 기록되며 해당 패널티 만큼 시간을 차단할 수 있다. ACC 내 Policy테이블의 구조와 설명은 아래와 같다.

Name	Type	Description
ID	String	랜덤으로 생성한 UUID
subjectId	String	네트워크의 ID (LookUpTable과 매칭되는)
Object	Object	네트워크 사용자의 이름과 맥주소를 가지는 객체
resource	String	사용자가 접근할 수 있는 네트워크의 자원
action	String	사용자가 접근할 수 있는 네트워크의 자원에 대한 행동 (C, R, U, D)
permission	Bool	사용자의 네트워크 접근 권한
toLR	String	사용자가 마지막으로 네트워크에 접근한 시간
timeOfUnblock	String	사용자가 비정상적인 접근 요청으로 접근 제한이 걸렸을 때 제한이 해제되는 시간
minInterval	Integer	네트워크 재요청시 최소로 필요한 시간(단위는 분)
noFR	Integer	minInterval 시간 이내에 요청된 접근 수
threshold	Integer	네트워크를 차단하기 위한 noFR의 임계치
misbehaviorTables	Array<Object>	네트워크 차단 시 차단 이유, 차단 패널티, 차단 발생 시간을 기록하는 배열

표 6 ACC 체인코드의 Policytable의 구조

ACC는 또한 오작동에 대해 기록을 하게 되는데 이를 위해 MisbehaviorTable을 사용하며 아래의 테이블과 같다.

Name	Type	Description
Misbehavior	String	오작동의 이름이다.
Time	String	오작동 발생 시간이다.
Penalty	Long	네트워크 사용자의 이름과 맥주소를 가지는 객체

표 7 ACC 체인코드의 MisbehaviorTable구조

4.2.3. Judge Contract(JC)

JC는 ACC로부터 잠재적인 오작동 보고서를 수신하여 해당 오작동을 판단하고 해당하는 페널티를 결정하는 오작동 판단 방법을 구현한다. 패널티는 오작동 내용에 근거한다. 패널티를 결정한 후에 JC는 결과를 ACC로 반환한다. ACC는 반환된 결과를 MisbehaviorTable에 기록한다.

4.3. 게이트웨이

게이트웨이는 앞서 설명한 바와 같이 PC와 스마트폰을 비롯한 다양한 IoT 장비들을 하나로 묶어 관리의 효율성을 증대시키기 위해 Raspberry Pi를 이용하여 구현하였으며 본 프로젝트에서 시연할 내용인 인터넷 자원을 제한하기 위해 리눅스에서 제공되는 기능인 hostapd와 masquerade를 이용하여 가상의 Access Point를 만들어 해당 Access Point에 접속하는 사용자들의 인터넷 자원을 제한한다. 또한 게이트웨이는 주기적으로 블록체인 네트워크와 통신하여 사용자들의 목록과 각 사용자들의 권한을 확인한 후 권한이 있는 사용자에 한해서만 인터넷 자원을 사용할 수 있도록 하는 기능을 구현하기 위해 iptables 라이브러리를 이용하였다.

4.4. FrontendServer

Frontend Server는 관리자가 쉽게 NAC Service를 관리하기 위해 구현하였다. 로그인, 채널 생성, 공유 네트워크 관리 페이지를 구현했으며 각 페이지에 필요한 정보들은 Frontend Server에서 Backend Server로 Rest API를 요청 하여 주고받는다. node express로 웹 어플리케이션 서버를 구현하였으며 내부 페이지는 express 템플릿 엔진(html 템플릿 언어)인 pug로 구현하였다. WebPack을 사용하여 모듈화된 javascript를 하나의 script파일로 컴파일하여 페이지 내에서 여러 script를 불러올 필요 없이 컴파일된 하나의 script를 불러오도록 하였다.

4.4.1. 페이지 구조

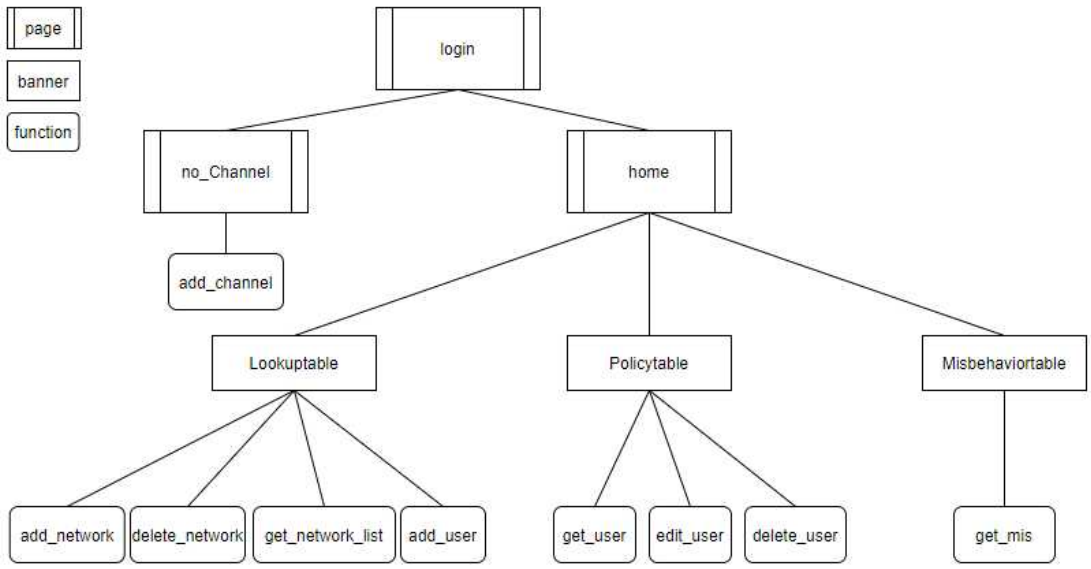

그림 6 페이지 구조도

Frontend Server의 페이지 구조는 위의 페이지 구성도와 같다. 페이지는 login, no_Channel, home 3가지이다. login은 블록체인 네트워크의 조직 관리자 권한을 얻기 위

해 로그인 기능을 제공한다. no_Channel은 로그인하였으나 블록체인 네트워크에 채널이 생성되지 않았을 때 나타나며 채널을 새로 생성하고 체인코드를 업로드 하는 기능을 제공한다. 로그인에 성공하고 채널이 정상적으로 생성됐다면 home 페이지로 이동한다. home 페이지는 Lookuptable, Policytable, Misbehavortable의 배너를 출력하며 채널 생성, 네트워크 생성(추가), 네트워크 삭제, 네트워크 리스트 보기, 사용자 추가, 사용자 삭제, 사용자 수정, 사용자 정보 보기, 오작동 보기 기능들로 구성되어 있다.

4.5. BackendServer

Fabric Client SDK를 사용하여 Fabric Network와 통신을 하며 Spring Boot를 사용하여 Rest API로 Frontend Server와 Gateway 간의 통신을 할 수 있게 구현하였다. Frontend Server와 gateway에서 Backend Server로 요청 API를 보내면 내부 Fabric Client가 Fabric Network에 체인코드를 호출하게 되고 결과 값을 가공하여 다시 Frontend Server와 gateway로 반환하도록 구현하였다. Fabric Client SDK를 이용하여 체인코드 호출뿐만 아니라 자신의 채널을 생성하고 관리자가 직접 체인코드를 올릴 수 있도록 하였다. 전체적인 기능으로는 관리자인지 확인하는 로그인과 체인코드 업로드, 기타 나머지 체인코드 호출기능을 한다.

5. 구현 및 알고리즘

5.1. 스마트컨트랙트

스마트컨트랙트는 3개의 체인코드로 구성되며 모두 java로 코딩되어있다. 체인코드를 동작하기 위한 DB는 CouchDB, NoSql로 구성되어 있다.

5.1.1. RC

스마트컨트랙트의 RC부분의 API 구현부분을 기술한다.

5.1.1.1. RC API목록

API Name	methodRegister		
Process	블록체인 네트워크에 공유할 네트워크 정보를 등록한다.		
Request	Name	Type	Description
	Id	String	랜덤으로 생성한 UUID
	methodName	String	네트워크를 설명할 이름
	subject	Object	게이트 웨이의 이름과 맥주소를 가지는 객체
	scName	String	Acc 체인코드의 이름
	abi	String	Acc 체인코드에서 실행할 접근 제어 함수 이름
Sucess Response	id	String	생성된 네트워크의 ID를 반환
Error Responce	반환값 없음		

표 8 methodRegister

API Name	methodUpdate		
Process	블록체인 네트워크에 공유할 네트워크 정보를 수정한다.		
Request	Name	Type	Description
	Id	String	등록된 네트워크가 가지고 있는 UUID
	methodName	String	네트워크를 설명할 이름
	subjectName	String	게이트 웨이의 이름
	objects	Array<Object>	네트워크 사용자의 이름과 맥주소를 가지는 객체의 배열의
	scName	String	Acc 체인코드의 이름
	abi	String	Acc 체인코드에서 실행할 접근 제어 함수 이름
Sucess Response	반환값 없음		
Error Responce	반환값 없음		

표 9 methodUpdate

API Name	methodDelete		
Process	블록체인 네트워크 내 공유되는 네트워크 정보를 삭제한다.		
Request	Name	Type	Description
	Id	String	등록된 네트워크가 가지고 있는 UUID
Sucess Response	반환값 없음		
Error Responce	반환값 없음		

표 10 methodDelete

API Name	getContract		
Process	블록체인 네트워크에서 해당 ID에 대한 네트워크 정보를 조회한다.		
Request	Name	Type	Description
	Id	String	등록된 네트워크가 가지고 있는 UUID
Sucess Response	methodName	String	네트워크를 설명하는 이름
	subject	Object	게이트웨이의 이름과 맥 주소를 포함하는 객체
	objects	Array <Object>	네트워크를 이용하는 사용자들의 이름과 맥 주소를 포함하는 객체
	scName	String	Acc 체인코드의 이름
	abi	String	Acc 체인코드에서 실행할 접근 제어 함수 이름
Error Responce	반환값 없음		

표 11 getContract

API Name	getMethodNameList		
Process	블록체인 네트워크에서 해당 체널이 가지고 있는 네트워크의 모든 아이디를 가지고 온다.		
Request	Name	Type	Description
	없음		
Sucess Response	networks	Array<Object>	네트워크의 이름과 id를 반환하는 배열
Error Responce	반환값 없음		

표 12 getMethodNameList

API Name	getObjList		
Process	블록체인 네트워크 내 등록된 네트워크에서 매개변수로 받은 Subject Mac Address와 일치하는 네트워크에서 ObjectList를 가져온다.		
Request	Name	Type	Description
	macAddress	String	네트워크 공유를 위해 등록해둔 게이트웨이의 맥 주소
Sucess Response	Objects	Array<Object>	네트워크의 등록된 사용자(object) 리스트를 반환. 등록된 사용자 이름과 맥 주소가 포함.
Error Responce	반환값 없음		

표 13 getObjList

5.1.2. ACC

스마트 컨트랙트의 ACC 체인코드의 구현 내용을 기술한다. 아래는 해당 체인코드의 알고리즘 흐름도이다.

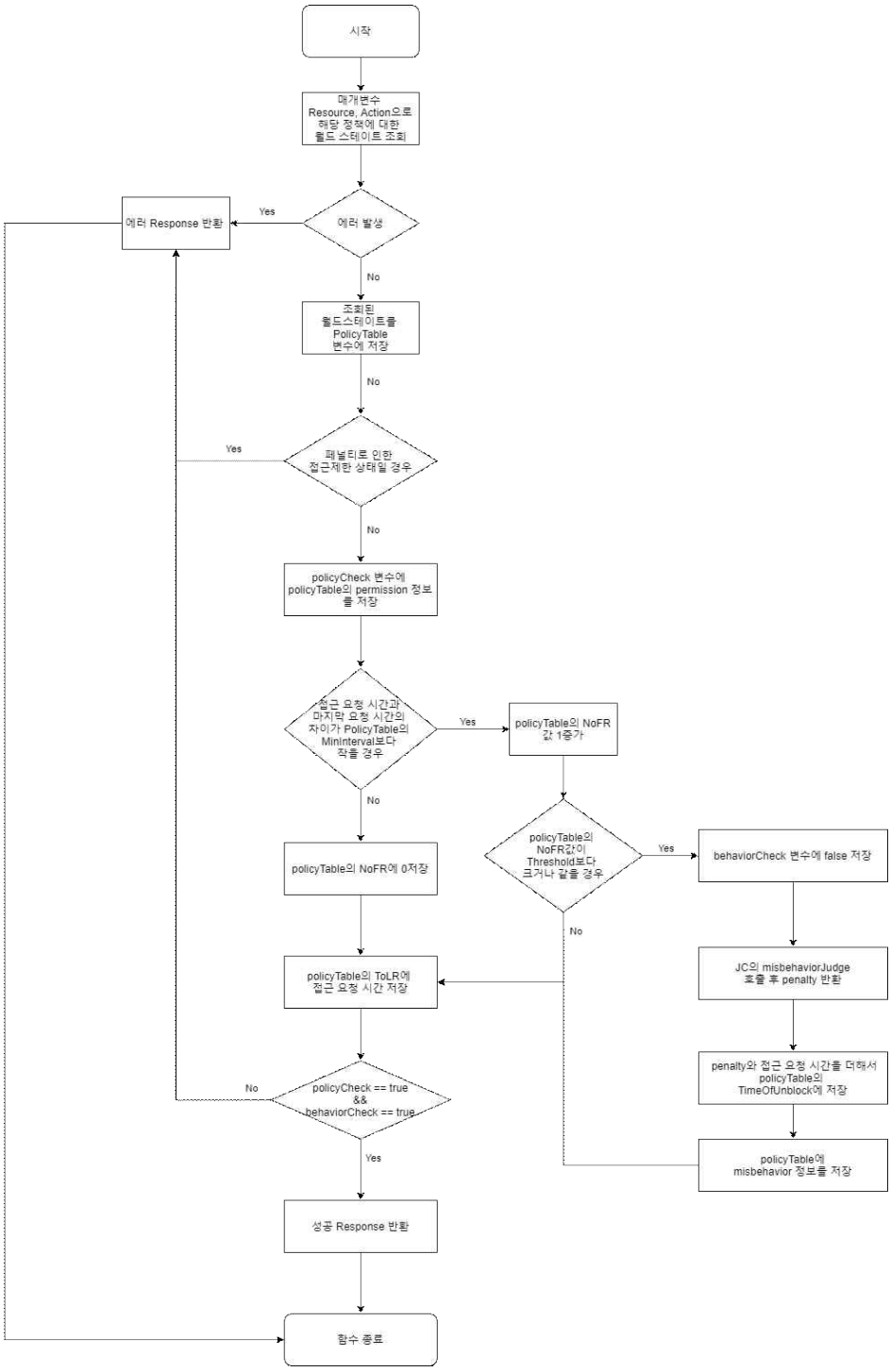

그림 7 ACC의 알고리즘 흐름도

5.1.2.1. ACC API 목록

API Name	policyAdd		
Process	사용자의 접근 권한 정책들을 추가한다.		
Request	Name	Type	Description
	Id	String	랜덤으로 생성한 UUID
	subjectgId	String	블록체인 네트워크에 등록된 네트워크의 ID
	object	Object	사용자의 이름과 맥주소를 가지는 객체
	resource	String	사용자가 접근할 수 있는 자원
	action	String	사용자가 접근할 수 있는 자원에 대한 행동
	permission	Bool	사용자의 네트워크 접근 권한
	threshold	Integer	사용자의 네트워크 차단을 위한 noFR의 임계치
	minInterval	Integer	사용자가 네트워크에 재요청할 수 있는 시간
Sucess Response	id	String	생성된 네트워크의 ID를 반환
Error Responce	반환값 없음		

표 14 policyAdd

API Name	getPolicy		
Process	사용자의 접근 권한 정책을 조회한다.		
Request	Name	Type	Description
	id	String	생성된 네트워크의 ID를 반환
Sucess Response	subjectId	String	네트워크의 ID (LookUpTable과 매칭되는)
	Object	Object	네트워크 사용자의 이름과 맥주소를 가지는 객체
	resource	String	사용자가 접근할 수 있는 네트워크의 자원
	action	String	사용자가 접근할 수 있는 네트워크의 자원에 대한 행동
	permission	Bool	사용자의 네트워크 접근 권한
	toLR	String	사용자가 마지막으로 네트워크에 접근한 시간
	timeOfUnblock	String	사용자가 비정상적인 접근 요청으로 접근 제한이 걸렸을 때 제한이 해제되는 시간
	minInterval	Integer	네트워크 재요청시 최소로 필요한 시간(단위는 분)
	noFR	Integer	minInterval 시간 이내에 요청된 접근 수
	threshold	Integer	네트워크를 차단하기 위한 noFR의 임계치
	misbehaviorTables	Array<Object>	네트워크 차단 시 차단 이유, 차단 패널티, 차단 발생 시간을 기록하는 배열
Error Responce	반환값 없음		

표 15 getPolicy

API Name	policyUpdate		
Process	사용자의 접근 권한 정책들을 수정한다.		
Request	Name	Type	Description
	Id	String	등록된 사용자가 가지고 있는 UUID
	objectName	String	수정할 사용자의 이름
	resource	String	사용자가 접근할 수 있는 자원
	action	String	사용자가 접근할 수 있는 자원에 대한 행동
	permission	Bool	사용자의 네트워크 접근 권한
	threshold	Integer	사용자의 네트워크 차단을 위한 noFR의 임계치
	minInterval	Integer	사용자가 네트워크에 재요청할 수 있는 시간
Sucess Response	반환값 없음		
Error Responce	반환값 없음		

표 16 policyUpdate

API Name	policyDelete		
Process	사용자의 접근 권한 정책을 삭제한다.		
Request	Name	Type	Description
	Id	String	등록된 사용자가 가지고 있는 UUID
Sucess Response	반환값 없음		
Error Responce	반환값 없음		

표 17 policyDelete

API Name	AccessControl		
Process	사용자가 네트워크에 접근 요청을 한다.		
Request	Name	Type	Description
	Id	String	등록된 사용자가 가지고 있는 UUID
	requestTime	String	네트워크에 접근 요청한 시간
Sucess Response	result	Bool	네트워크의 접근이 허가되면 true, 불허되면 false를 반환
Error Responce	반환값 없음		

표 18 AccessControl

5.1.3. JCC

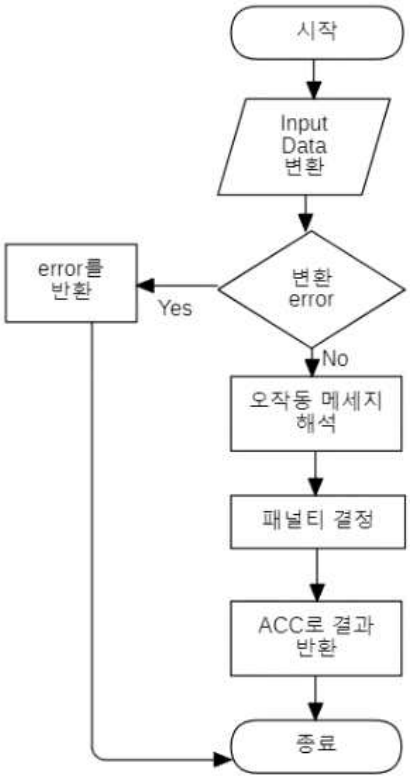

그림 8 JC의 주요 알고리즘 흐름도

JC는 ACC로부터 오작동이 탐지되었을 때 해당 오작동의 페널티를 결정하여 ACC로 반환하는 체인코드이다. 해당 동작의 알고리즘은 위의 JC의 주요 알고리즘 흐름도와 같다.

5.1.3.1. JC API 목록

API Name	misbehaviorJudge		
Process	오작동 원인을 매개변수로 받은 후 오작동에 해당하는 패널티를 acc로 반환한다.		
Request	Name	Type	Description
	msb	String	오작동 원인
	msbLen	String	해당 사용자가 오작동을 일으킨 횟수
Sucess Response	penalty	String	패널티 시간 (분)
Error Responce	반환값 없음		

표 19 misbehaviorJudge

5.2. 게이트웨이

5.2.1. 게이트웨이 함수 목록

Name	IoTGateway.__init__
Input	networkInterface, webServerAddress, macAddress, webServerPort
Process	코드 실행 시 사용자로부터 입력받은 인자와 netifaces 라이브러리를 통해 확인한 네트워크 인터페이스의 물리적 주소를 클래스의 멤버 변수로 초기화 한다.
Output	없음

표 20 IoTGateway.__init__

Name	IoTGateway.run
Input	없음
Process	IoTGateway 클래스를 동작시키기 위한 함수로 특정 노드의 차단과 허용을 위해 python-iptables 라이브러리를 사용하여 규칙과 체인을 생성하며 IoTGateway의 다양한 메소드를 호출하여 의도한 기능을 수행할 수 있도록 한다.
Output	없음

표 21 IoTGateway.run

Name	IoTGateway.resetChain
Input	inputChain, forwardChain
Process	python-iptables 라이브러리를 이용하여 특정 노드를 차단 혹은 허용하는 체인을 생성할 때마다 버퍼에 해당 체인의 목록이 누적되는데 이로 인해 의도치 않은 동작이 수행되는 것을 피하기 위해 주기적으로 호출되어 inputChain과 forwardChain을 초기화하는 동작을 수행한다.
Output	없음

표 22 IoTGateway.resetChain

Name	IoTGateway.setDrop
Input	inputChain, forwardChain
Process	해당 함수에서는 inputChain과 forwardChain에 대해 DROP을 설정하는 동작을 수행한다. DROP 대상은 inputChain의 TCP와 UDP 프로토콜 그리고 forwardChain의 모든 패킷이며 사용자가 와이파이에 접속을 시도할 때 IP 주소 할당을 위해 사용되는 DHCP는 허용하여야 하므로 해당 프로토콜의 포트인 67번 포트에 대해서는 DROP을 설정하지 않는다. 또한 python-iptables에서는 체인의 등록 순서에 대한 우선순위가 존재하는데 늦게 설정된 체인일수록 우선순위가 높기 때문에 setDrop 함수가 setAccept 함수보다 먼저 실행되어야 원활한 동작을 수행할 수 있다.
Output	없음

표 23 IoTGateway.setDrop

Name	IoTGateway.getNetworkMap
Input	없음
Process	해당 함수는 게이트웨이에서 현재 블록체인 네트워크에 등록된 사용자들의 리스트를 얻기 위해 주기적으로 호출되는 함수이며 IoTGateway 클래스의 멤버 변수로 가지고 있던 웹 서버의 주소와 포트 그리고 게이트웨이 네트워크 인터페이스의 물리적 주소를 이용하여 요청 URL과 요구되는 헤더를 설정한 후 블록체인 네트워크로 사용자 리스트를 요청한다.
Output	동작에 성공할 시 json으로 변환된 사용자 리스트 반환

표 24 IoTGateway.getNetworkMap

Name	IoTGateway.getPermission
Input	networkMap
Process	해당 함수는 IoTGateway.getNetworkMap의 반환값으로 얻은 사용자 리스트에 대해 웹 서버의 주소와 포트 그리고 해당 사용자의 uuid를 이용하여 블록체인 네트워크에 해당 사용자의 접근 권한을 질의한다. 블록체인 네트워크는 해당 질의에 대한 응답으로 특정 사용자의 권한을 반환하는 것이 아닌 HTTP의 응답 상태 코드를 이용하여 해당 사용자의 권한을 확인할 수 있도록 한다.
Output	없음

표 25 IoTGateway.getPermission

Name	IoTGateway.setAccept
Input	inputChain, forwardChain
Process	해당 함수는 IoTGateway의 멤버 변수로 존재하는 사용자 리스트에 대한 inputChain과 forwardChain의 ACCEPT 설정을 담당하는데 사용자 리스트를 순회하며 권한이 있는 사용자들에 대해서만 물리적 주소 기반의 ACCEPT를 설정함으로써 특정 사용자의 네트워크 접근을 허용할 수 있도록 한다. 또한 이 함수는 블록체인 네트워크의 사용자 리스트에 변동이 있을 때만 실행되도록 하며 체인 생성 시 버퍼에 누적된다는 점과 python-iptables에 우선순위가 존재한다는 점을 고려하여 resetChain과 setDrop 함수가 먼저 실행된 후 실행되어야 정상 동작을 수행할 수 있다.
Output	없음

표 26 IoTGateway.setAccept

Name	IoTGateway.printChain
Input	없음
Process	해당 함수는 관리자 혹은 개발자가 현재 게이트웨이의 상태를 확인하고 문제 발생 시 디버깅을 용이하도록 하기 위해 콘솔 메시지를 이용하여 정보를 제공한다.
Output	없음

표 27 IoTGateway.printChain

5.2.2. 게이트웨이 자료 구조

Key	데이터 목록	자료형	요약
IoTGateway	networkMap	List	사용자 리스트
	networkInterface	String	게이트웨이 네트워크 인터페이스
	webServer	String	웹 서버 주소
	webPort	String	웹 서버 포트
	macAddress	String	네트워크 인터페이스 물리적 주소
networkMap	macAddress	String	등록된 사용자의 물리적 주소
	uuid	String	등록된 사용자의 uuid
	permission	Boolean	등록된 사용자의 권한

표 28 게이트웨이 자료구조 목록

5.2.3. 게이트웨이 플로우 차트

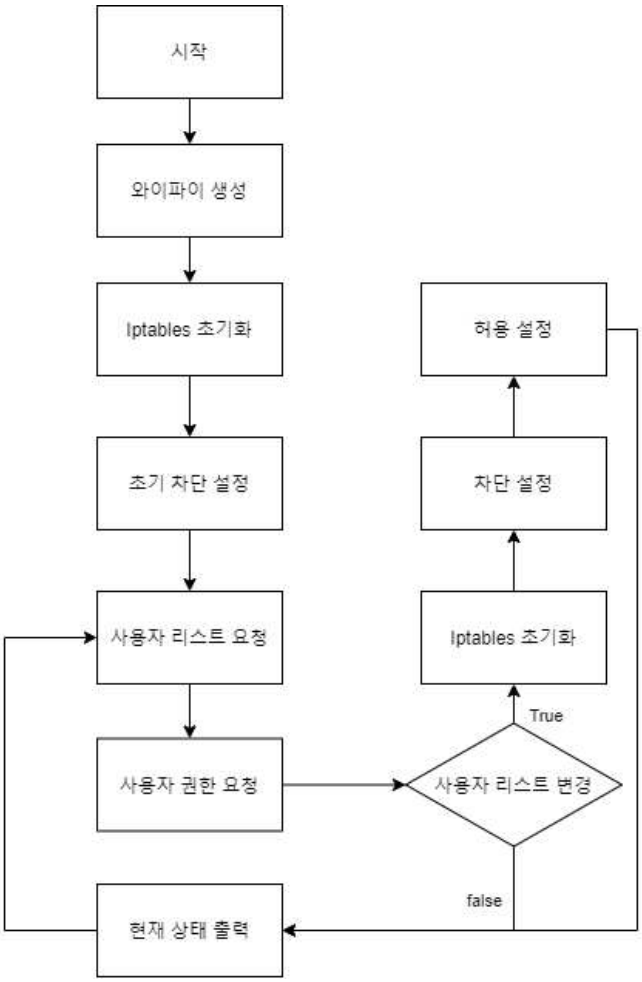

그림 9 게이트웨이 플로우 차트

5.3. Fronted Server

FrontedServer에서 구현된 login, no_channel, home 페이지들과, LookUpTable, PolicyTable, MisbehaviorTable등의 배너 들의 구현 모습을 설명하고 네트워크 추가, 네트워크 삭제, 네트워크 리스트 보기, 사용자 추가, 사용자 삭제, 사용자 정보 보기, 사용자 수정, 오작동 보기 등의 기능이 프런트에서 구현된 모습을 설명한다.

5.3.1. login

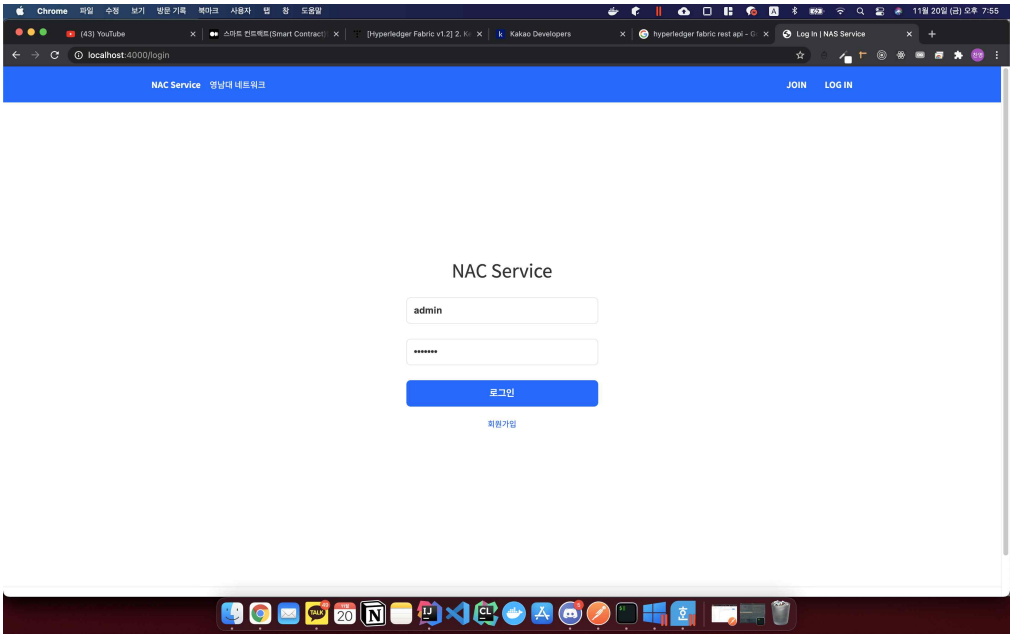

그림 10 login 페이지

로그인 페이지 위의 그림과 같고 기본적인 로그인, 회원가입 기능을 제공한다.

5.3.2. no_channel

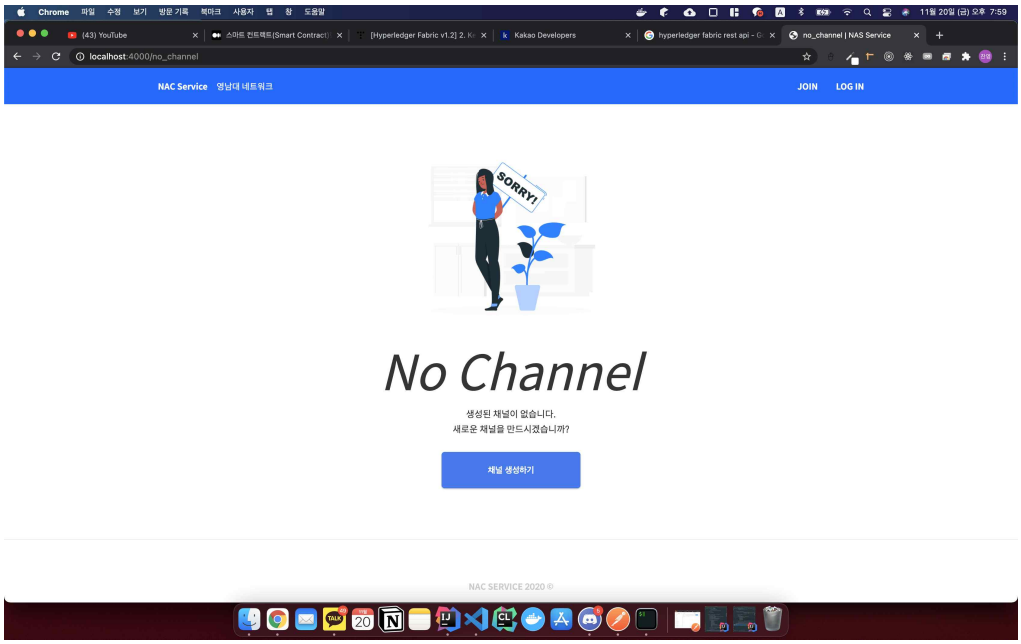

그림 11 no_channel 페이지

로그인하였으나 채널이 생성되지 않았을 때 보이는 페이지다. 해당 페이지에서 채널 생성을 생성할 수 있다. 채널 생성하기 버튼을 누르면 아래의 그림들처럼 체인코드가 올라가면서 채널이 생성된다.

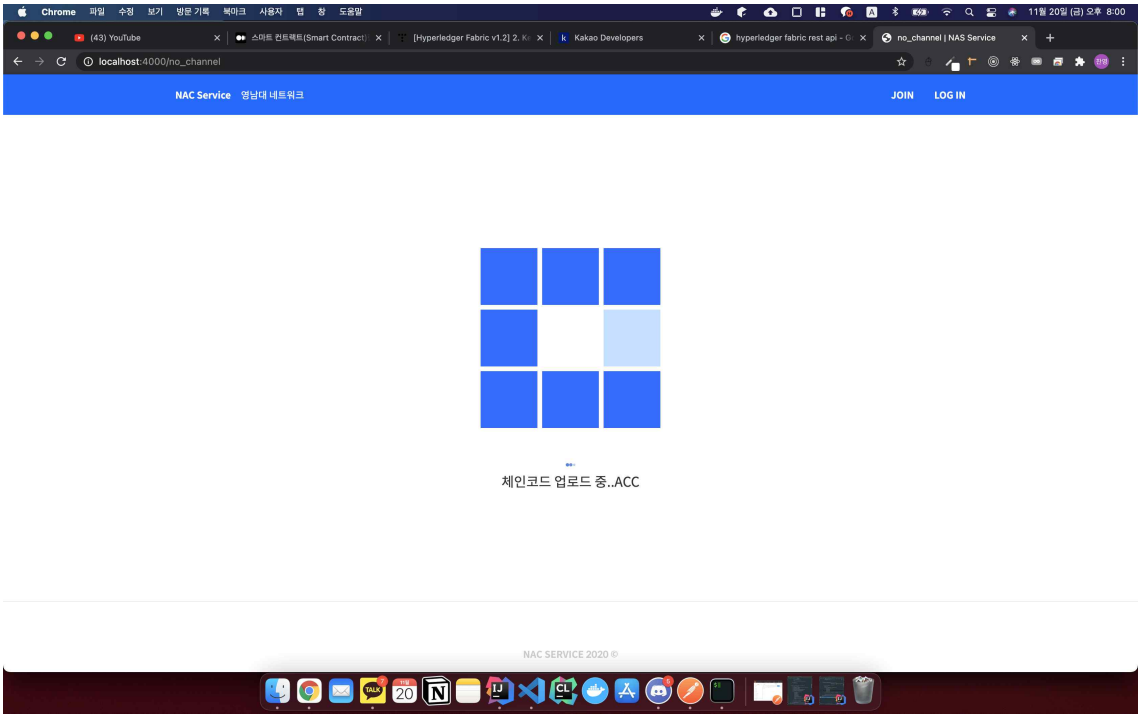

그림 12 Chaincode Uploading

5.3.3. home

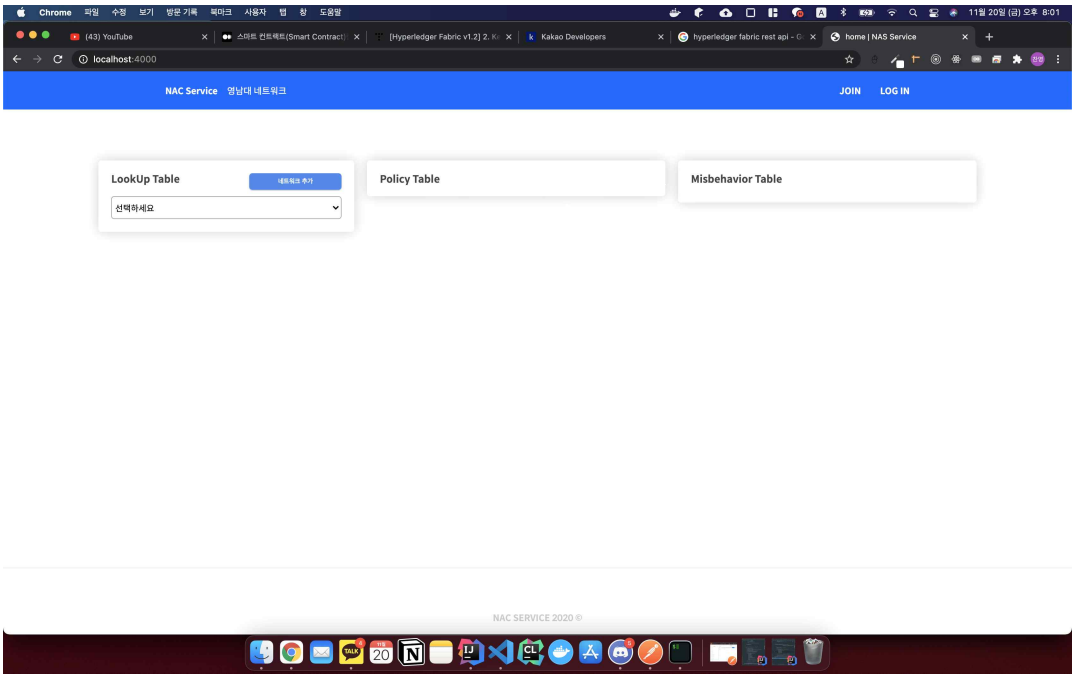

그림 13 home 페이지

로그인과 채널 생성을 성공적으로 마치면 위의 그림처럼 home 페이지에 접근할 수 있다. 홈페이지는 LookUpTable, PolicyTable, Misbehavior 배너로 구성되어 있고 이 3가지 배너를 통해 네트워크 생성, 네트워크 추가, 네트워크 삭제, 사용자 추가, 사용자 삭제, 오작동 탐지, 네트워크 리스트 확인 및 패널티를 확인을 할 수 있다.

5.3.4. 네트워크 생성

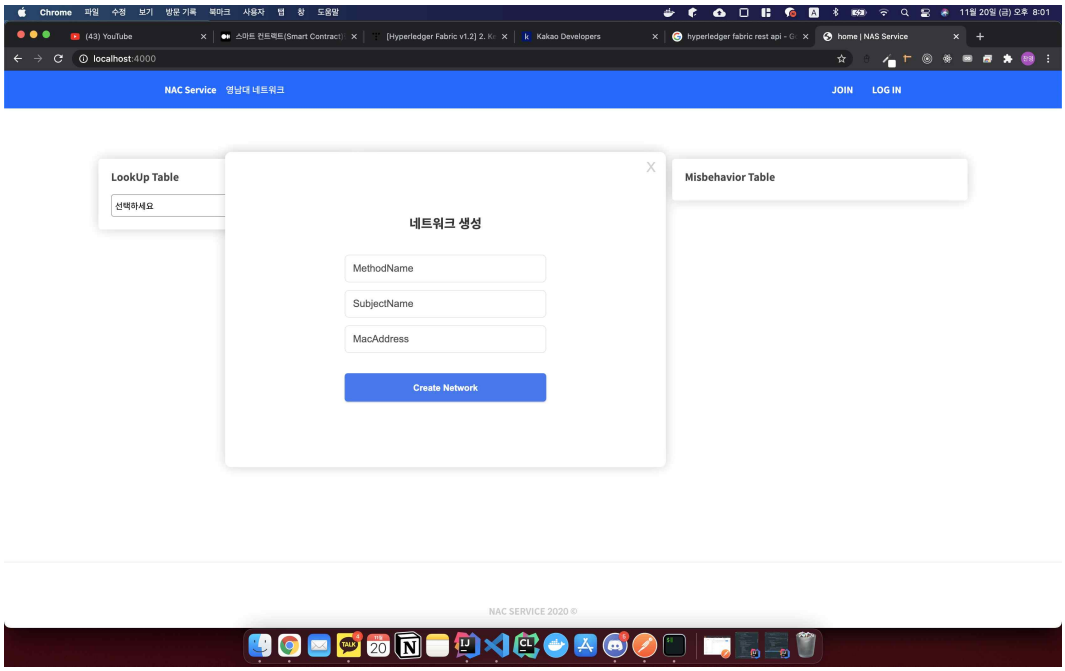

그림 14 네트워크 생성

home 페이지의 LookUpTable배너에서 네트워크 생성을 클릭하면 위의 그림처럼 네트워크 생성 팝업창이 뜨게 된다. 해단 팝업창에서 MethodName, SubjectName, MacAddress를 입력하여 네트워크를 생성한다. 네트워크를 생성하면 아래와 같은 그림의 화면으로 페이지가 나타나게 된다. 추가적인 네트워크 생성도 위의 내용의 순서를 반복하면 새로운 네트워크 생성된다.

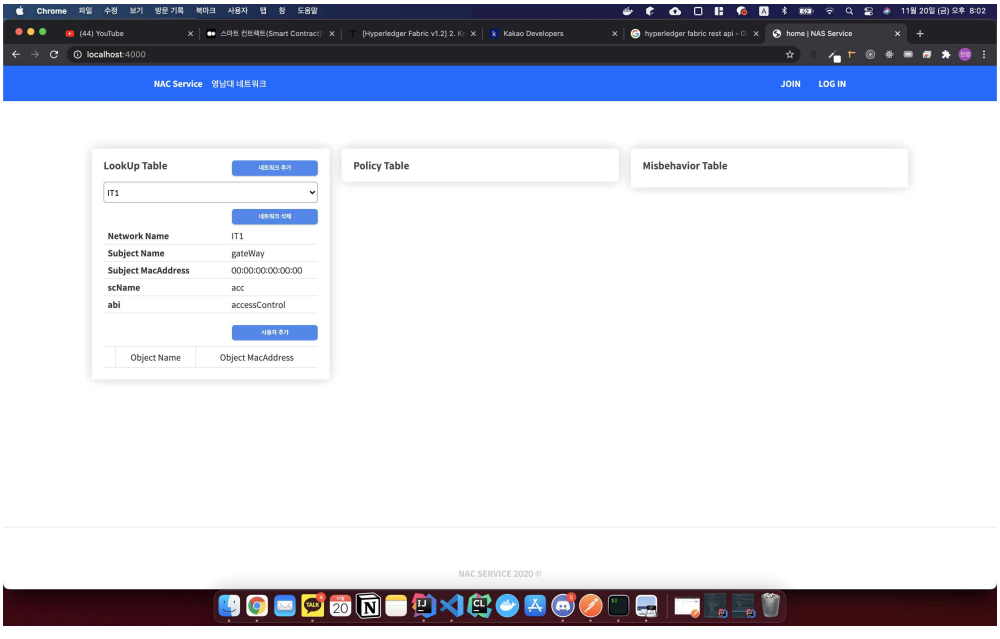

그림 15 네트워크를 생성후의 모습

5.3.5. 네트워크 삭제

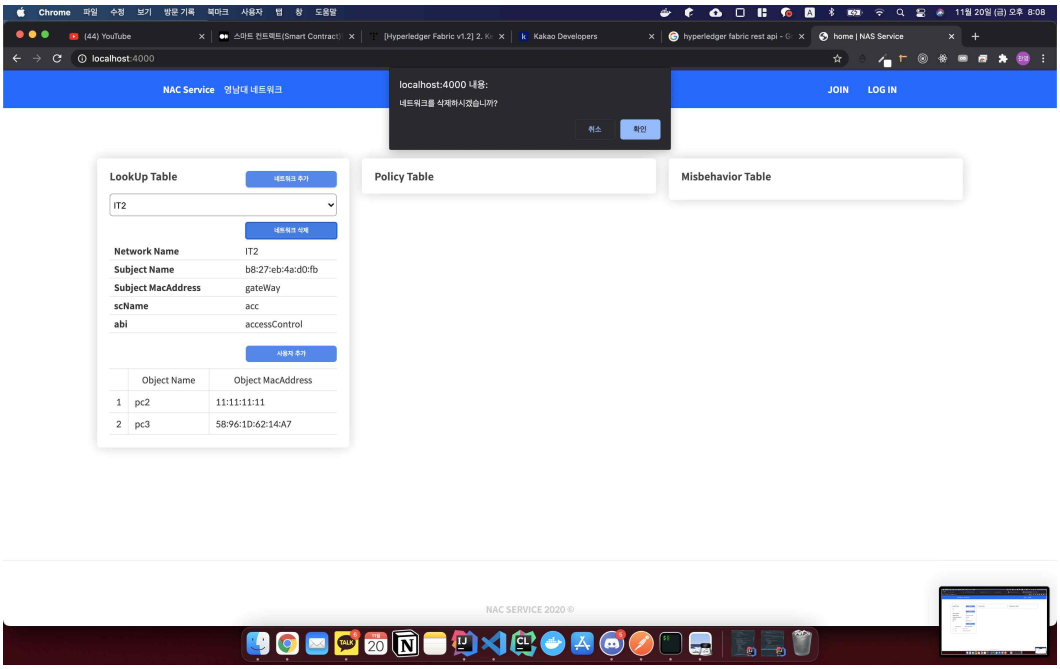

그림 16 네트워크 삭제

home 페이지의 LookUpTable 배너의 네트워크 리스트에서 삭제하고 싶은 네트워크를 선택한 후 네트워크 삭제 버튼을 누르면 작은 팝업창이 뜨면서 네트워크를 삭제하게 된다. 네트워크 삭제가 이루어진 후의 모습은 아래의 그림과 같다.

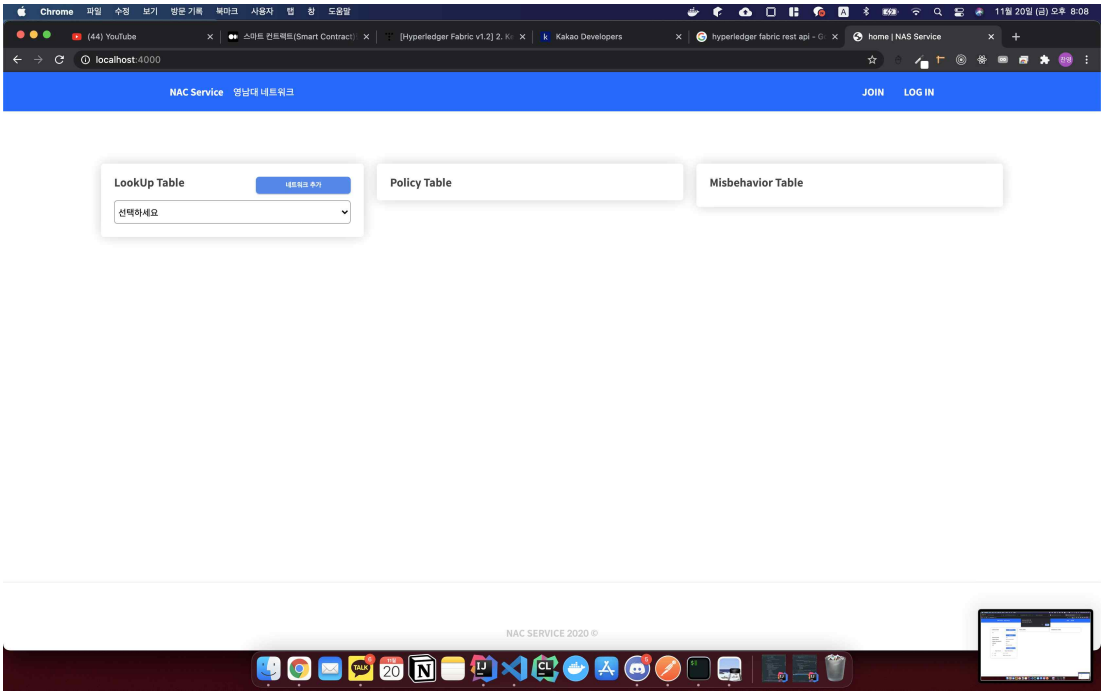

그림 17 네트워크가 삭제된 모습

5.3.6. 네트워크 리스트 보기

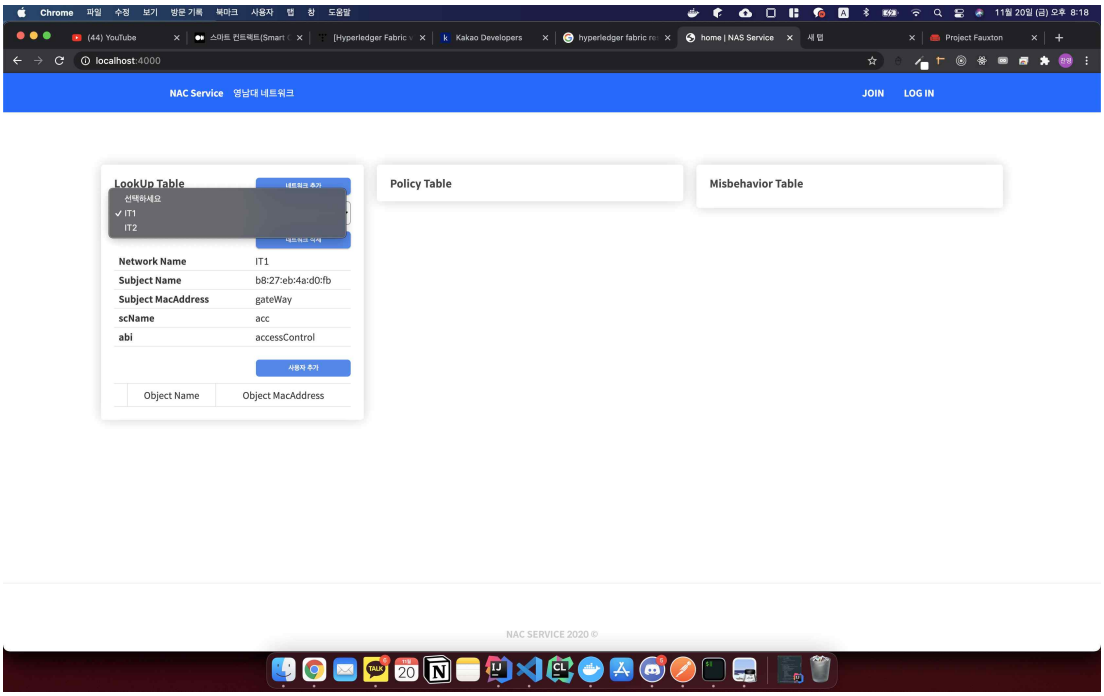

그림 18 네트워크 리스트

home 페이지의 LookUpTable 배너에서 네트워크 리스트를 클릭하게 되면 위의 그림처럼 네트워크 목록이 나오게 되며 체크박스를 선택 시 해당 네트워크 관리 페이지로 넘어갈 수 있다.

5.3.7. 사용자 정보 확인

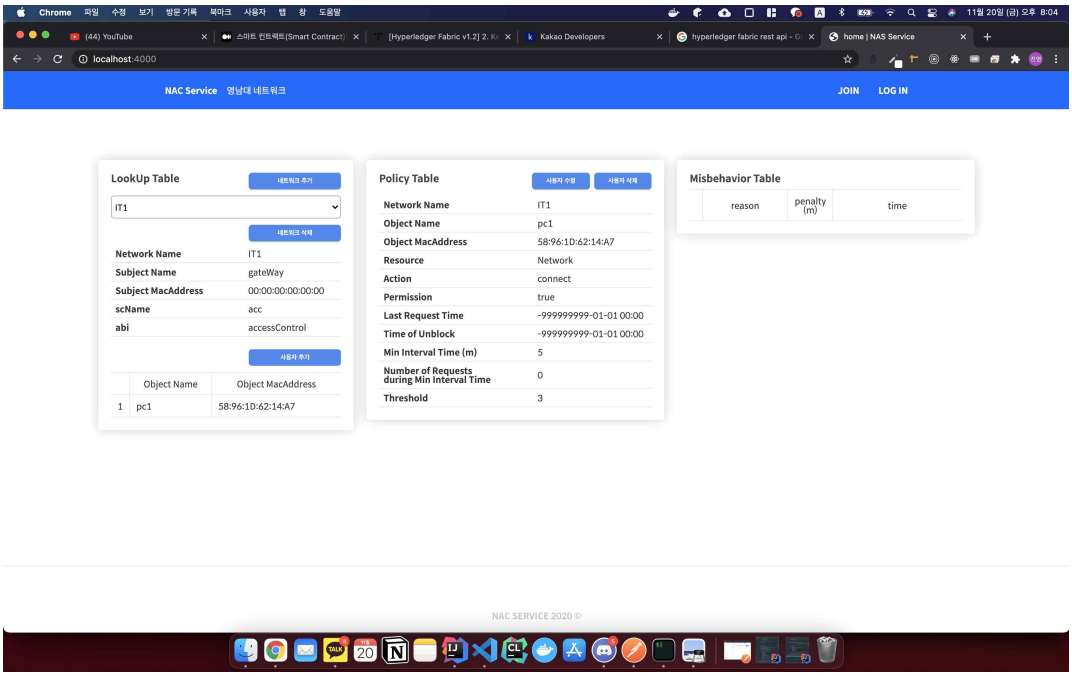

그림 19 사용자 정보

LookUptable배너의 아래쪽에 사용자 목록에서 사용자 튜플을 클릭하게 되면 PolicyTable의 사용자 정보를 볼 수 있다. 사용자의 정보의 모습은 위의 그림과 같다.

5.3.8. 사용자 추가

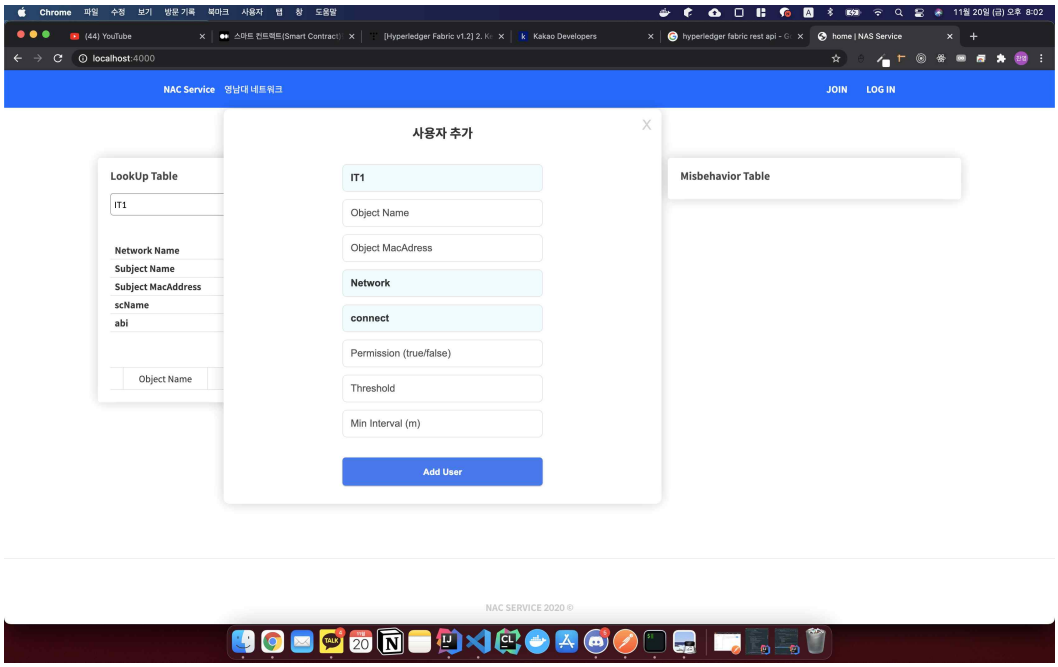

그림 20 사용자 추가

Lookuptable에서 사용자 추가 버튼을 누르게 되면 위의 그림처럼 사용자 추가 팝업이 나오게 된다. 이 팝에 ObjectName, ObjectMacAdress, Permission, Threshold, Min Interval 등을 입력하여 사용자를 추가하게 된다. 사용자를 추가하면 아래의 그림의 모습으로 나타나게 되고 새로운 사용자를 추가할 때도 위와 같은 방법을 반복하면 된다.

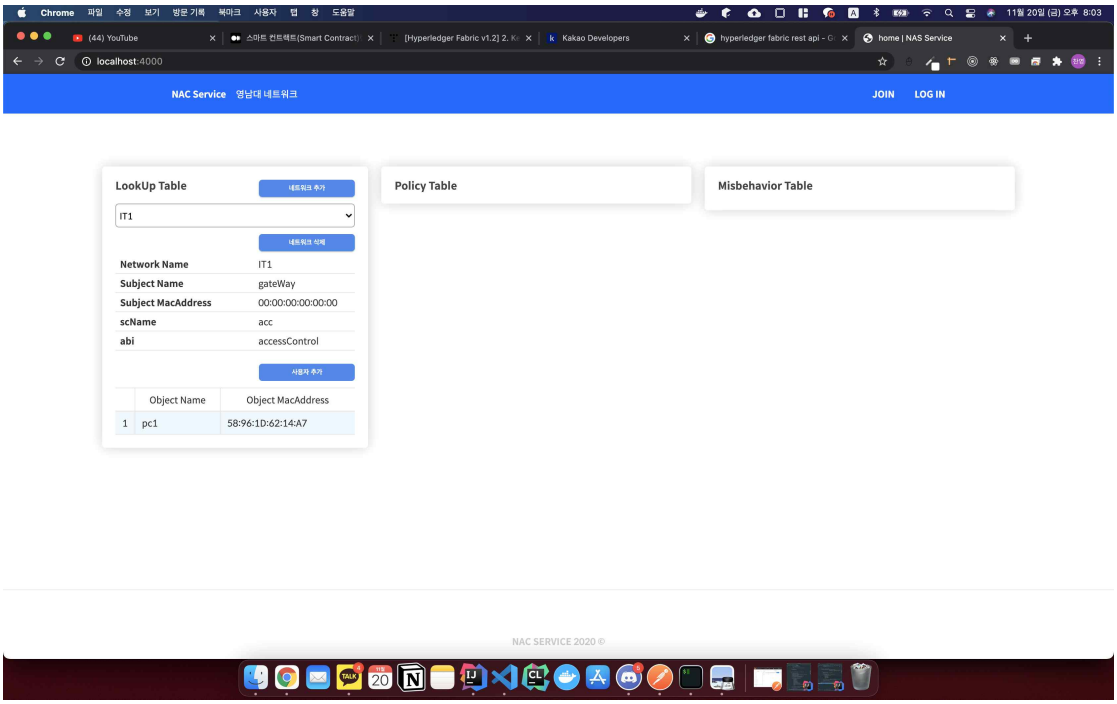

그림 21 사용자가 추가된 모습

5.3.9. 사용자 삭제

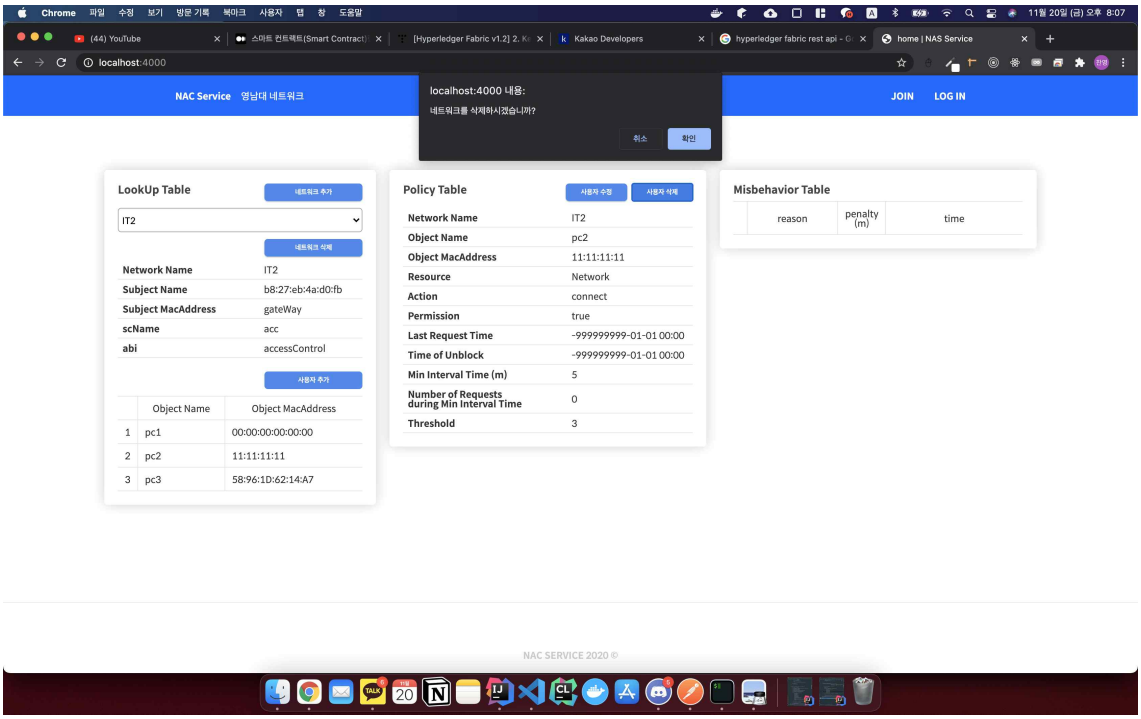

그림 22 사용자 삭제

PolicyTable의 사용자 삭제를 눌러 해당 사용자의 정보를 삭제할 수 있다. 사용자를 삭제한 후의 모습은 아래의 그림과 같다.

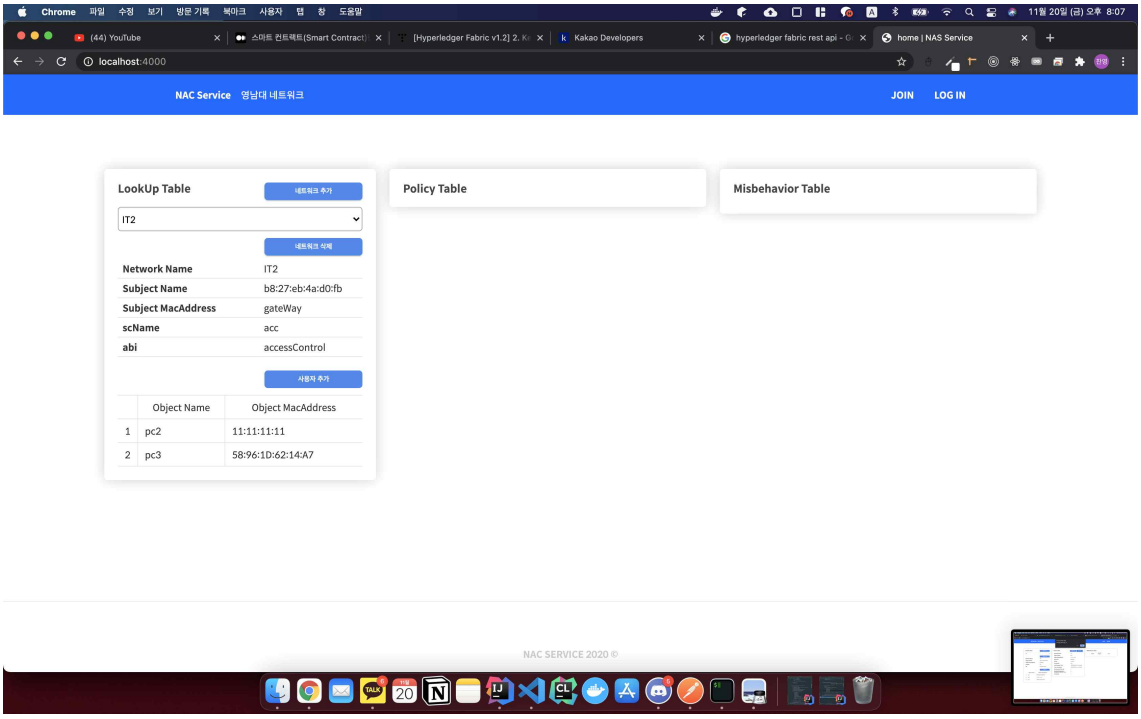

그림 23 사용자를 삭제한 후의 모습

5.3.10. 사용자 수정

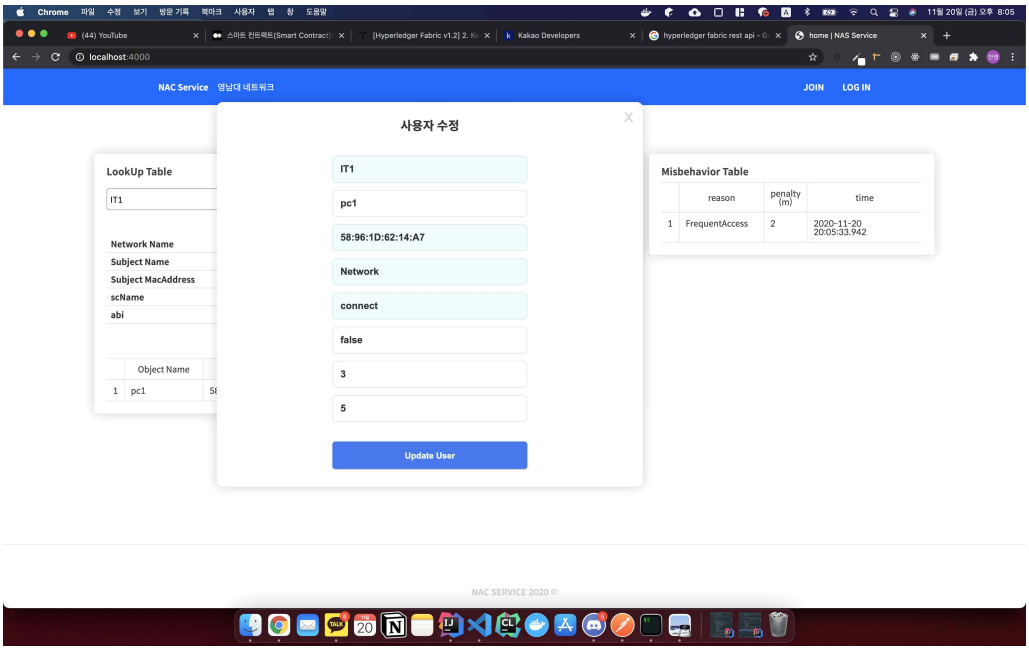

그림 24 사용자 수정

PolicyTable의 사용자 수정 버튼을 누르면 사용자 수정 팝업창이 뜨면서 사용자의 정보, 상태를 수정할 수 있다.

5.3.11. 오작동 확인

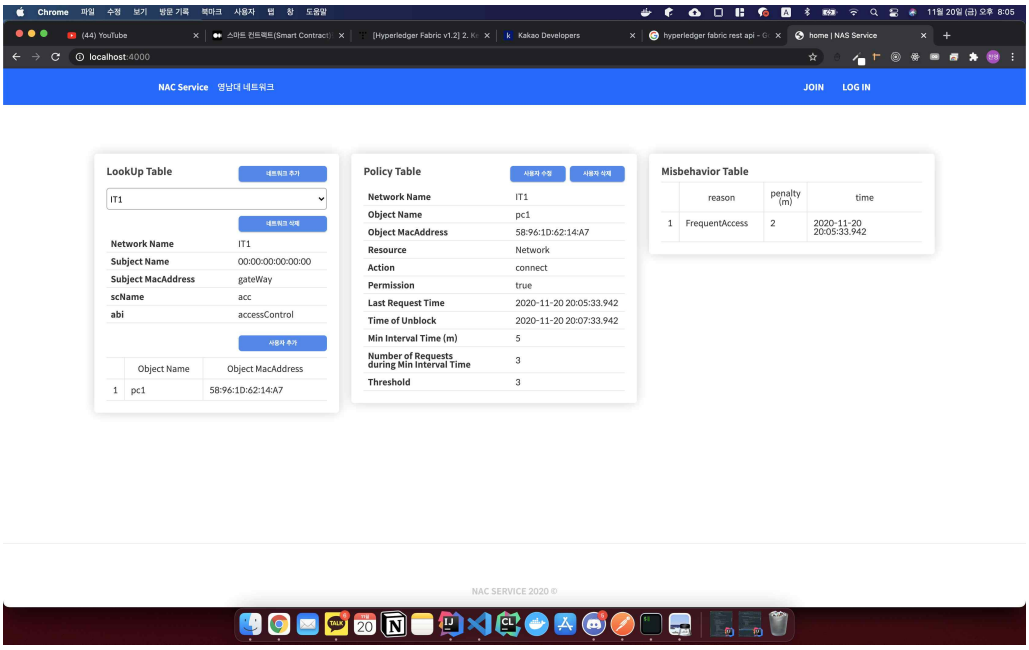

그림 25 오작동 확인

오작동이 있을 시 MisbehaviorTable에서 해당 오작동 내용을 위의 그림처럼 확인할 수 있다.

5.4. BackendServer

해당 장에서 BackendServer에서 구현된 Rest API들을 설명한다.

5.4.1. Rest API

API Name	Login		
Process	사용자 권한을 얻기 위해 아이디 비밀번호로 로그인.		
URL	POST host/api/v1/NACUser/login header : 'Content-Type' : 'application/Json'		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	Id	String	사용자 id
	Pw	String	사용자 비밀번호
Success Response	반환값 없음		
Error Response	반환값 없음		

표 29 Login

API Name	isCreatedChannel		
Process	로그인한 조직의 채널이 생성되었는지 확인.		
URL	GET host/api/v1/NACUser/isCreatedChannel header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	result	Bool	채널이 있다면 true, 없다면 false
Error Response	반환값 없음		

표 30 isCreatedChannel

API Name	createChannel		
Process	로그인한 조직이 생성된 채널이 없다면 채널을 생성한다.		
URL	GET host/api/v1/NACUser/createChannel header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	반환값 없음		
Error Response	반환값 없음		

표 31 createChannel

API Name	depolyRC		
Process	해당 채널에 RC 체인코드를 등록한다.		
URL	GET host/api/v1/NACUser/deployRC header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	반환값 없음		
Error Response	반환값 없음		

표 32 depolyRC

API Name	depolyACC		
Process	해당 채널에 ACC 체인코드를 등록한다.		
URL	GET host/api/v1/NACUser/deployACC header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	반환값 없음		
Error Response	반환값 없음		

표 33 depolyACC

API Name	depolyJC		
Process	해당 채널에 JC 체인코드를 등록한다.		
URL	GET host/api/v1/NACUser/deployJC header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	반환값 없음		
Error Response	반환값 없음		

표 34 depolyJC

API Name	addLookUpTable		
Process	공유할 네트워크를 추가하는 RC의 addLookUpTable 체인코드를 호출 후 결과를 반환한다.		
URL	POST host/api/v1/lookUpTables header : { 'Content-Type' : 'application/Json' 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK header : { 'Location' : '/lookUpTables/UUID'}	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	methodName	String	네트워크를 설명할 이름
	subject	Object	게이트 웨이의 이름과 맥주소를 가지는 객체
Success Response	반환값 없음		
Error Response	반환값 없음		

표 35 addLookUpTable

<table>
<tr><td>API Name</td><td colspan="3">updateLookUpTable</td></tr>
<tr><td>Process</td><td colspan="3">공유할 네트워크를 수정하는 RC의 updateLookUpTable 체인코드를 호출 후 결과를 반환한다.</td></tr>
<tr><td>URL</td><td colspan="3">PUT host/api/v1/lookUpTables/{id}
header : { ‘Content-Type’ : ‘application/Json’
‘orgMspId’ : ‘’
‘orgAffiliation’ : ‘’ }</td></tr>
<tr><td rowspan="2">Response State</td><td>Success</td><td colspan="2">HTTP/1.1 200 OK</td></tr>
<tr><td>Error</td><td colspan="2">HTTP/1.1 422 UNPROCESSABLE_ENTITY</td></tr>
<tr><td rowspan="8">Request</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>id</td><td>String</td><td>등록된 네트워크가 가지고 있는 UUID</td></tr>
<tr><td>methodName</td><td>String</td><td>네트워크를 설명할 이름</td></tr>
<tr><td>objects</td><td>Array<Object></td><td>네트워크 사용자의 이름과 맥주소를 가지는 객체의 배열의</td></tr>
<tr><td>subjectName</td><td>String</td><td>게이트 웨이의 이름</td></tr>
<tr><td>scName</td><td>String</td><td>Acc 체인코드의 이름</td></tr>
<tr><td>abi</td><td>String</td><td>Acc 체인코드에서 실행할 접근 제어 함수 이름</td></tr>
<tr><td>Success Response</td><td colspan="3">반환값 없음</td></tr>
<tr><td>Error Response</td><td colspan="3">반환값 없음</td></tr>
</table>

표 36 updateLookUpTable

API Name	deleteLookUpTable		
Process	공유할 네트워크를 삭제하는 RC의 deleteLookUpTable 체인코드를 호출 후 결과를 반환한다.		
URL	DELETE host/api/v1/lookUpTables/{id} header : { ‘orgMspId’ : ‘’ ‘orgAffiliation’ : ‘’ }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	Id	String	등록된 네트워크가 가지고 있는 UUID
Success Response	반환값 없음		
Error Response	반환값 없음		

표 37 deleteLookUpTable

API Name	getContractList		
Process	해당 채널이 가지고 있는 모든 네트워크를 조회하는 RC의 getContractList 체인코드를 호출 후 결과를 반환한다.		
URL	GET host/api/v1/lookUpTables header : { ‘orgMspId’ : ‘’ ‘orgAffiliation’ : ‘’ }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	없음		
Success Response	networks	Array<Object>	네트워크의 이름과 id를 반환하는 배열
Error Response	반환값 없음		

표 38 getContractList

API Name	getContract		
Process	네트워크를 조회하는 RC의 getContract 체인코드를 호출 후 결과를 반환한다.		
URL	GET host/api/v1/lookUpTables/{id} header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	Id	String	등록된 네트워크가 가지고 있는 UUID
Success Response	methodName	String	네트워크를 설명하는 이름
	subject	Object	게이트웨이의 이름과 맥 주소를 포함하는 객체
	objects	Array <Object>	네트워크를 이용하는 사용자들의 이름과 맥 주소를 포함하는 객체
	scName	String	Acc 체인코드의 이름
	abi	String	Acc 체인코드에서 실행할 접근 제어 함수 이름
Error Response	반환값 없음		

표 39 getContract

API Name	getObjList		
Process	매개 변수로 받은 macAddress와 일치하는 네트워크의 모든 사용자를 조회하는 RC의 getObjList 체인코드를 호출 후 결과를 반환한다.		
URL	GET host/api/v1/lookUpTables header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	macAddress	String	네트워크 공유를 위해 등록해둔 게이트웨이의 맥 주소
Success Response	Objects	Array<Object>	네트워크의 등록된 사용자(object) 리스트를 반환. 등록된 사용자 이름과 맥 주소가 포함.
Error Response	반환값 없음		

표 40 getObjList

<table>
<tr><td>API Name</td><td colspan="3">policyAdd</td></tr>
<tr><td>Process</td><td colspan="3">사용자의 접근 권한 정책을 추가하는 ACC의 policyAdd 체인코드를 호출 후 결과를 반환한다.</td></tr>
<tr><td>URL</td><td colspan="3">POST host/api/v1/policyTables
header : { ‘Content-Type’ : ‘application/Json’
‘orgMspId’ : ‘’
‘orgAffiliation’ : ‘’ }</td></tr>
<tr><td rowspan="2">Response State</td><td>Success</td><td colspan="2">HTTP/1.1 200 OK
header : { ‘Location’ : ‘/lookUpTables/UUID’}</td></tr>
<tr><td>Error</td><td colspan="2">HTTP/1.1 422 UNPROCESSABLE_ENTITY</td></tr>
<tr><td rowspan="9">Request</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>subjectgId</td><td>String</td><td>블록체인 네트워크에 등록된 네트워크의 ID</td></tr>
<tr><td>object</td><td>Object</td><td>사용자의 이름과 맥 주소를 가지는 객체</td></tr>
<tr><td>resource</td><td>String</td><td>사용자가 접근할 수 있는 자원</td></tr>
<tr><td>action</td><td>String</td><td>사용자가 접근할 수 있는 자원에 대한 행동</td></tr>
<tr><td>permission</td><td>Bool</td><td>사용자의 네트워크 접근 권한</td></tr>
<tr><td>threshold</td><td>Integer</td><td>사용자의 네트워크 차단을 위한 noFR의 임계치</td></tr>
<tr><td>minInterval</td><td>Integer</td><td>사용자가 네트워크에 재요청할 수 있는 시간</td></tr>
<tr><td>Success Response</td><td colspan="3">반환값 없음</td></tr>
<tr><td>Error Response</td><td colspan="3">반환값 없음</td></tr>
</table>

표 41 policyAdd

API Name	getPolicy		
Process	사용자의 접근 권한 정책을 조회하는 ACC의 getPolicy 체인코드를 호출 후 결과를 반환한다.		
URL	GET host/api/v1/policyTables/{id} header : { 'orgMspId' : '' 'orgAffiliation' : '' }		
Response State	Success	HTTP/1.1 200 OK header : { 'Location' : '/lookUpTables/UUID'}	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	id	String	등록된 사용자가 가지고 있는 UUID
Success Response	subjectId	String	네트워크의 ID (LookUpTable과 매칭되는)
	Object	Object	네트워크 사용자의 이름과 맥 주소를 가지는 객체
	resource	String	사용자가 접근할 수 있는 네트워크의 자원
	action	String	사용자가 접근할 수 있는 네트워크의 자원에 대한 행동
	permission	Bool	사용자의 네트워크 접근 권한
	toLR	String	사용자가 마지막으로 네트워크에 접근한 시간
	timeOfUnblock	String	사용자가 비정상적인 접근 요청으로 접근 제한이 걸렸을 때 제한이 해제되는 시간
	minInterval	Integer	네트워크 재요청시 최소로 필요한 시간(단위는 분)
	noFR	Integer	minInterval 시간 이내에 요청된 접근 수
	threshold	Integer	네트워크를 차단하기 위한 noFR의 임계치
	misbehaviorTables	Array<Object>	네트워크 차단 시 차단 이유, 차단 패널티, 차단 발생 시간을 기록하는 배열
Error Response	반환값 없음		

표 42 getPolicy

<table>
<tr><td>API Name</td><td colspan="3">policyUpdate</td></tr>
<tr><td>Process</td><td colspan="3">사용자의 접근 권한 정책을 수정하는 ACC의 policyUpdate 체인코드를 호출 후 결과를 반환한다.</td></tr>
<tr><td>URL</td><td colspan="3">PUT host/api/v1/policyTables/{id}
header : { ‘Content-Type’ : ‘application/Json’
‘orgMspId’ : ‘’
‘orgAffiliation’ : ‘’ }</td></tr>
<tr><td rowspan="2">Response State</td><td>Success</td><td colspan="2">HTTP/1.1 200 OK</td></tr>
<tr><td>Error</td><td colspan="2">HTTP/1.1 422 UNPROCESSABLE_ENTITY</td></tr>
<tr><td rowspan="8">Request</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>Id</td><td>String</td><td>등록된 사용자가 가지고 있는 UUID</td></tr>
<tr><td>objectName</td><td>String</td><td>수정할 사용자의 이름</td></tr>
<tr><td>resource</td><td>String</td><td>사용자가 접근할 수 있는 자원</td></tr>
<tr><td>action</td><td>String</td><td>사용자가 접근할 수 있는 자원에 대한 행동</td></tr>
<tr><td>permission</td><td>Bool</td><td>사용자의 네트워크 접근 권한</td></tr>
<tr><td>threshold</td><td>Integer</td><td>사용자의 네트워크 차단을 위한 noFR의 임계치</td></tr>
<tr><td>minInterval</td><td>Integer</td><td>사용자가 네트워크에 재요청할 수 있는 시간</td></tr>
<tr><td>Success Response</td><td colspan="3">반환값 없음</td></tr>
<tr><td>Error Response</td><td colspan="3">반환값 없음</td></tr>
</table>

표 43 policyUpdate

API Name	policyDelete		
Process	사용자의 접근 권한 정책을 수정하는 ACC의 policyDelete 체인코드를 호출 후 결과를 반환한다.		
URL	DELETE host/api/v1/policyTables/{id} header : { ‘orgMspId’ : ‘’ ‘orgAffiliation’ : ‘’ }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	Id	String	등록된 사용자가 가지고 있는 UUID
Success Response	반환값 없음		
Error Response	반환값 없음		

표 44 policyDelete

API Name	AccessControl		
Process	사용자 접근 요청을 하는 ACC의 accessControl 체인코드를 호출 후 결과를 반환한다. 접근 권한 승인시 상태 200을 반환하고 실패시 422를 반환한다.		
URL	GET host/api/v1/accessControl header : { ‘orgMspId’ : ‘’ ‘orgAffiliation’ : ‘’ }		
Response State	Success	HTTP/1.1 200 OK	
	Error	HTTP/1.1 422 UNPROCESSABLE_ENTITY	
Request	Name	Type	Description
	Id	String	등록된 사용자가 가지고 있는 UUID
Success Response	반환값 없음		
Error Response	반환값 없음		

표 45 AccessControl

5.5. 소스코드

해당 구현된 시스템의 소스코드는 아래의 링크로 제공한다.

https://github.com/yuackr/blockchainNAC

6. 테스트 방법과 결과

기능	테스트 내용	구현 상태
정책 등록	권한이 있는 사용자에 의해서만 정책이 등록되는가	√
	올바른 입력 값에만 동작하는가	√
	사용자에게 제공된 UI를 통해서만 동작하는가	√
정책 갱신	권한이 있는 사용자에 의해서만 정책이 갱신되는가	√
	올바른 입력 값에만 동작하는가	√
	사용자에게 제공된 UI를 통해서만 동작하는가	√
정책 삭제	권한이 있는 사용자에 의해서만 정책이 삭제되는가	√
	사용자에게 제공된 UI를 통해서만 동작하는가	√
권한 등록	권한이 있는 사용자에 의해서만 권한이 등록되는가	√
	올바른 입력 값에만 동작하는가	√
	사용자에게 제공된 UI를 통해서만 동작하는가	√
권한 갱신	권한이 있는 사용자에 의해서만 권한이 갱신되는가	√
	올바른 입력 값에만 동작하는가	√
	사용자에게 제공된 UI를 통해서만 동작하는가	√
접근 통제	접근 권한을 가지고 있는 사용자에 대해 접근을 허용하는가	√
	접근 권한이 없는 사용자에 대해 접근을 차단하는가	√
이상 행동 탐지	등록된 정책에 위배되는 행위를 탐지할 수 있는가	√
	등록된 정책을 위반하는 사용자에게 패널티를 부과할 수 있는가	√
	패널티가 기준을 초과한 사용자에 대해 접근을 차단할 수 있는가	√
IoT 및 기기 연결	여러 통신 장비와 IoT 장비들을 블록체인 네트워크에 연결시킬 수 있는가	√
	접근이 허용된 IoT 장비에 대해 데이터 통신을 중계할 수 있는가	√
	일정 수 이상의 IoT 장비와 연결되었을 때 지연시간을 적정 수준 이하로 유지할 수 있는가	√
	스레드를 이용하여 여러 IoT 장비의 데이터를 병렬 처리할 수 있는가	√
User Interface	사용자의 권한에 맞게 다른 화면을 출력할 수 있는가	√
	사용자의 요청에 대한 응답을 잘 처리할 수 있는가	√

표 46 테스트 체크 리스트 표.

7. 개발 환경 및 테스트 환경

7.1. 개발 환경

Block Chain
- HyperLedger Fabric : 1.4.1

Chain Code (RC, ACC, JC)
- 사용 언어 : JAVA
- Fabirc Chain Code JAVA SDK : 1.4.1

Gateway
사용언어 Python

Backend Server
- 사용 언어 : JAVA
- Spring Boot : 2.3.5
- Fabric Client SDK : 1.4.1

Frontend Server
- 사용 언어 : JavaScript, scss, pug(html)
- Node Express : 4.17.1
- WebPack : 4.43.0
- Pug : 3.0.0

7.2. 테스트 환경

Hyperledger Fabric Network
- Docker

Frontend Server
mac osx Big sur, windows 10

Backend Server
mac osx Big sur, windows 10

Gateway
하드웨어 : 라즈베리 파이
Raspbian OS 5.4

8. 결론 및 향후 연구

해당 보고서의 내용을 통해 블록체인 기반의 네트워크 접근 제어 시스템을 성공적으로 완성 할 수 있었다. 기존의 중앙형 네트워크 접근제어에서 벗어나 블록체인 기반의 네트워크 접근제어 시스템을 구현함으로써 보안성, 분산성, 무결성, 경제성을 또한 Iot장비들의 접근제어를 가능하게 하였다는것에 의의를 가진다. 이외에도 시스템의 코어인 블록체인 네트워크와 스마트 컨트랙트는 네트워크 자원에 한정된 것이 아닌 파일, 기기들의 동작들 여러 가지 자원의 접근 관리를 할 수 있는 범용성을 지녔음으로 향후 더욱 연구와 개발을 거듭하여 다양한 자원에 대응 할 수 있는 네트워크 접근 시스템을 구현할 수 있을 것이다.

9. 참고 문헌

- Smart Contract-Based AccessControl for the Internet of Things - Yuanyu Zhang, Member, IEEE, Shoji Kasahara, Member, IEEE, Yulong Shen, Member, IEEE,
- Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT - Oscar Novo
- 블록체인 비즈니스를 위한 하이퍼레저 패브릭 - Nakul Shah
- hyperledger fabric sdk java - https://github.com/hyperledger/fabric-sdk-java
- hyperledger fabric Introduction - https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html
- Fabric CA User's Guide - https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html#fabric-ca-user-s-guide
- ldx / python-iptables - https://github.com/ldx/python-iptables