JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 3, pp. 73-80, March 2020
https://doi.org/10.9708/jksci.2020.25.03.073

A robust execution scheme for Ethereum blockchain application services

Min-Ho Kwon*, Myung-Joon Lee*

*Student, Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea
*Professor, Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea

[Abstract]

In this paper, we propose a robust execution scheme for Ethereum blockchain application services which provide continuous services and support recovery even in the case of failures of those services. As of now, blockchain services are rapidly expanding to all industries and being combined with various infrastructure services of public institutions. But if these services fail in disastrous situations such as earthquakes, floods and terrors or various hacking attacks, the aftermath is very serious and its side effects are difficult to assess. To resolve this issue, we develop the service registry as a smart contract of the Ethereum blockchain, which provides useful information on the services to users even in disasters. With the help of the service registry, we also present a method of replicating services on the basis of the ETCD distributed storage system. The proposed scheme is confirmed by the test service developed by the proposed techniques.

▸ **Key words**: Blockchain Service, Service Robustness, Service Replication, Service Recovery, Service Registry

[요 약]

본 논문에서는 이더리움 블록체인 응용 서비스의 실패에도 지속적인 서비스 제공 및 복구를 지원하는 견고한 실행 기법에 대해 제안한다. 현재 블록체인 응용서비스는 전 산업으로 급속히 확장되고 있으며 공공 기관의 다양한 인프라 형태의 서비스와 결합하고 있다. 그러나 만약 지진, 홍수 및 테러와 같은 재난이나 다양한 해킹 공격과 같은 상황에서 이러한 서비스가 실패하게 될 경우 그 여파는 매우 심각하며 그 부작용을 가늠하기 어렵다. 이를 해결하기 위하여 재난 상태에서도 사용자에게 응용 서비스에 대하여 유용한 정보를 제공하는 서비스 레지스트리를 블록체인의 스마트 컨트랙트로 개발하고, 이를 기반으로 분산 저장소인 ETCD를 이용한 서비스 복제 기법을 제시하며, 제시한 기법을 시범적인 응용 서비스에 적용하여 그 유용성을 확인한다.

▸ **주제어**: 블록체인 서비스, 서비스 견고성, 서비스 복제, 서비스 복원, 서비스 레지스트리

• First Author: Min-Ho Kwon, Corresponding Author: Myung-Joon Lee
*Min-Ho Kwon (alsgh458@gmail,com), Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan
*Myung-Joon Lee (mjlee@ulsan.ac.kr), Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan
• Received: 2020. 02. 26, Revised: 2020. 03. 16, Accepted: 2020. 03. 16.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

위변조가 사실상 불가능한 것으로 인정되고 프로그래밍이 가능한 블록체인 기술의 등장으로 빠르게 확산되는 블록체인의 활용이 전 산업 분야로 확장되고 있다.[1] 특히 정부의 각종 증명서와 계약, 표결과 같은 디지털화된 기록[2-4] 및 공공 기관, 금융[5], 의학, 행정의 인프라 같은 데이터의 신뢰성이 중요한 영역에서는 블록체인 응용 서비스가 더욱 빠르게 개발되고 있다.[6-8]

그러나 만약 대규모 인프라나 클라우드, 의료[9], 금융와 관련된 블록체인의 응용 서비스가 각종 재난과 해킹과 같은 상황에 의해 장애나 파괴로 서비스가 실패할 경우 막대한 비용이 발생하거나 그 부작용을 예측하기 어렵다. 현재 블록체인 응용서비스 실행의 견고성이나 실패에 대한 체계적인 연구는 활발히 이루어지고 있지 않은 상황이다.

본 논문은 이더리움 블록체인 응용 서비스의 견고한 실행 기법에 대해 제안한다. 블록체인 응용 서비스가 장애나 결함에도 지속적인 서비스를 제공하기 위하여, Raft(Reliable, Replicated, Redundant and Fault-Tolerant) 컨센서스[10] 프로토콜을 사용하는 분산 저장소인 ETCD(/etc, distributed)[19]를 이용한 서비스 복제 기법에 대하여 설명한다. 서비스 복제 기법은 분산 환경에서 실행되고 있는 각 노드에 응용 서비스를 복제하여 분산 실행하고 Raft 컨센서스 프로토콜을 사용하는 ETCD를 이용하여 복제된 응용 서비스의 상태를 지속적으로 동기화 및 복구를 지원한다. 또한 재난 상태에서도 서비스 복제 기법의 견고한 실행을 위하여 응용 서비스 정보를 관리하고 서비스 접근 위치 및 복구 요청 기능을 제공하는 서비스 레지스트리를 이더리움 블록체인의 스마트 컨트랙트[11-14]로 개발한다. 더불어 제안한 기법을 적용한 시범적인 응용 서비스를 개발하고 서비스 복제 기법의 테스트를 통해 그 유용성을 확인한다.

II. Background knowledge

1. Ethereum blockchain and Smart contract

블록체인은 분산 컴퓨팅 기술 기반의 데이터 위변조 방지 기술이다. P2P 방식을 기반으로 사용자의 요청에 해당하는 트랜잭션을 블록이라는 단위 데이터에 저장하며, 블록을 체인 형태로 연결하여 합의 알고리즘을 통해 분산 환경을 구성하는 노드에게 복제한다. 이러한 특성 때문에 블록체인에 저장된 데이터는 임의로 수정할 수 없으며 변경의 결과를 누구나 열람할 수 있다.

이더리움은 2세대 블록체인으로 스마트 계약 기능을 제공하기 위하여 고안된 블록체인 플랫폼이다. 이더리움 블록체인은 1세대 블록체인에 없는 반복문과 제어문을 사용할 수 있는 튜링 완전성을 가지고 있어 간단한 거래뿐만 아니라 복잡한 형태의 탈중앙화 애플케이션을 만들 수 있는 장점이 있다.

이더리움은 p2p 네트워크, 전자서명, 암호 해시 등 기본적인 기능을 제공한다. 또한 사용자에 의해 생성된 트랜잭션과 트랜잭션에 관련된 데이터들이 모여 있는 블록의 유효성을 검증하는 합의 엔진 등을 관리한다. 이더리움은 스마트 컨트랙트의 실행을 이더리움 가상 머신을 통하여 처리하며 스마트 컨트랙트의 각종 데이터 구조를 정의하고 관련된 데이터를 관리한다.

스마트 컨트랙트란 실제 사람이 대면하는 기존 계약 방식에서 벗어나 블록체인 상에서 조건에 따라 자동으로 수행하는 스마트 계약 기능이다. 스마트 컨트랙트는 이더리움 블록체인 기반에서 실행되기 때문에 무결성 보장 및 조작 방지가 가능한 실행 코드이다. 계약을 원하는 사용자는 배포된 스마트 컨트랙트를 이용하기 위해 트랜잭션을 보내고 스마트 컨트랙트에 정의된 내용에 따라 기능을 제공 받는다. 또한 스마트 컨트랙트에서는 기능을 수행하기 전후로 특정한 기능을 수행할 수 있도록 하는 함수 변경자, 트랜잭션 내에서 호출되어 특정 값을 클라이언트에 전달하고 트랜잭션에 로그 형태로 저장되는 이벤트 기능을 통해 스마트 컨트랙트의 기능은 안전하고 다양한 형태로 사용될 수 있다.

2. Raft

Raft는 분산 환경에서 참여 노드들이 충돌 오류와 같은 상황에도 복제 로그를 관리하기 위한 합의 알고리즘이다. Raft가 등장하기 이전에는 분산 환경에서 발생하는 합의 문제를 Paxos[22-23] 합의 알고리즘로 해결하였다. 그러나 Paxos는 합의 문제를 해결하는 과정이 길고 이해하기 힘든 단점이 존재한다. Paxos와 같은 성능을 내면서도 합의 문제를 이해하기 쉽게 해결하기 위하여 Raft가 개발되었다. Raft는 합의 그룹의 참여 노드 중 과반수에 장애가 발생하기 전까지 합의 그룹들은 정상적으로 동작함을 보장하며 분산 시스템의 검증 도구인 Verdi[16]를 통해 유효성을 검증받았다. 또한 Raft는 여러 산업에 사용되고 있는 블록체인 플랫폼인 하이퍼레저[17], 비즈니스를 위한 블록체인인 아르고[18], 신뢰성 있는 분산 저장소인 ETCD등 다양한 개발 환경[15]에서 사용되고 있다.

Raft는 분산 그룹의 참여 노드들이 발생하는 합의 문제를 이해하기 쉽게 해결하기 위하여 강력한 리더, 로그 관

리 및 안정성인 3가지로 분리하였다. Raft에서 참여 노드의 상태는 리더, 팔로워, 후보자인 3가지 상태가 있으며 상태 변이를 통해 분산 합의 그룹 간의 안정성을 유지한다. 또한 참여 노드들은 Raft 알고리즘에 의해 강력한 리더를 선출하여 분산 환경에서 동일한 상태를 가질 수 있도록 로그를 복사 및 관리 작업을 수행하도록 한다.

3. ETCD

ETCD는 분산 시스템 또는 시스템 클러스터에서 데이터를 안정적으로 저장할 수 있는 Raft 기반의 키-값 분산 저장소이다. 많은 개발 환경에서 ETCD는 간단한 웹 응용 서비스뿐만 아니라 도커[21] 등의 컨테이너에 대한 오케스트레이션 도구인 쿠버네티스[20] 같은 복잡한 응용 서비스에서도 사용되는 신뢰성 있는 분산 저장소이다. ETCD는 저장된 데이터 관찰 및 변화에 대한 이벤트 기능을 지원하여 저장된 데이터를 지속적으로 관리하는 데도 용이하다. 또한 ETCD는 분산 그룹 내에서 발생할 수 있는 실패에 대한 일관성 있는 정책과 복구 기능 및 사용자 인증서를 통한 자동 TLS(Transport Layer Security)를 제공하여 더욱 안전한 실행을 보장한다.

III. Robust execution scheme

본 장에서는 이더리움 블록체인 용용 서비스의 견고한 실행을 위한 서비스 복제 기법을 설명한다. 서비스 복제 기법은 분산 환경에서 응용 서비스를 복제하기 위하여, Raft 컨센서스 프로토콜을 사용한 분산 저장소인 ETCD를 이용한다. 또한 재난 상태에서도 서비스 복제 기법이 적용된 이더리움 블록체인 응용 서비스의 견고한 실행을 위하여, 서비스의 중요한 정보를 저장하고 서비스의 접근 위치 제공 및 복구 요청 기능을 제공하는 서비스 레지스트리를 스마트 컨트랙트로 개발한다.

1. Method of replicating services

제안하는 서비스 복제 기법은 블록체인 응용 서비스에 응용 서비스를 최소 3개 노드가 있는 분산 환경에서 복제한다. 각 노드에는 응용 서비스와 ETCD가 실행되고 있다.

Raft 컨센서스 프로토콜을 사용하는 ETCD는 분산 환경에서 로그 일치 작업을 수행하는 강력한 리더를 선출하여 분산 그룹들이 동일한 로그를 가질 수 있도록 한다. 해당 기법에서는 이러한 ETCD의 리더와 연결된 응용 서비스를 실행되고 있는 모든 응용 서비스에 리더로 선출한다. 리더 응용 서비스는 서비스 사용자에게 서비스를 제공하며 모든 응용 서비스의 상태를 동기화하기 위하여 사용자 요청을 ETCD에 저장한다. 그림 1은 서비스 복제 기법이 적용된 블록체인 응용 서비스를 나타낸다.

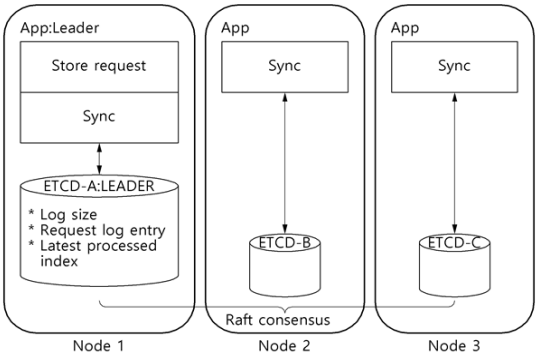

Fig. 1. Replication of Blockchain application

서비스 복제 기법에서 모든 응용 서비스의 상태를 동기화하고 서비스 실패를 복구하기 위하여 필요한 ETCD의 키-값 엔트리는 그림 1에 있는 Request Log Entry, Log Size, Latest Processed Index가 있다. Request Log Entry는 블록체인 응용 서비스의 사용자 요청 값을 인덱스 형태로 저장한 키-값 엔트리다. Log Size는 ETCD에 저장된 사용자 요청의 총 개수를 나타내는 키-값 엔트리다. Latest Processed Index는 각 노드에 실행되고 있는 응용 서비스가 상태를 동기화하기 위하여 최근에 사용한 Request Log Entry의 인덱스를 기록한 키-값 엔트리이다.

제안하는 서비스 복제 기법에서 작업은 크게 사용자 요청 저장과 동기화 작업이 있다. 사용자 요청 저장 작업은 리더 응용 서비스가 블록체인 응용 서비스의 사용자 요청을 ETCD에 저장하는 작업이다. 리더 응용 서비스는 사용자 요청을 받으면 처리 및 응답을 보류하고 즉시 사용자 요청을 ETCD에 저장한다. 리더 응용 서비스가 사용자 요청을 ETCD에 저장하는 과정은 다음과 같다.

(1) 요청에 대한 응답을 동기화 작업에서 수행하기 위하여 사용자가 보낸 요청에 현재 리더 응용 서비스의 고유한 식별 값을 저장하고 요청 커넥션을 커넥션 풀에 저장하여 응답을 보류한다.

(2) 사용자 요청을 직렬화 및 압축하여 ETCD에 저장할 수 있는 형태로 만든다.

(3) 사용자 요청을 저장할 Request Log Entry의 인덱스를 (현재 Log Size의 값 + 1)로 설정하고 ETCD에 사용자 요청을 저장한다.

(4) 저장이 완료되면 ETCD의 Log Size 값을 1 증가시켜 업데이트한다.

서비스 복제 기법에서 응용 서비스의 최신 상태는 응용 서비스가 ETCD에 현재 저장된 모든 Request Log Entry를 이용하였을 때 가지는 상태이다. 동기화는 각 노드에 실행되고 있거나 실패나 장애로 인해 재시작한 응용 서비스가 최신 상태를 따라가기 위한 작업이다. 응용 서비스의 동기화 과정은 다음과 같다.

(1) ETCD 내에서 Log Size의 업데이트 이벤트가 발생하여 응용 서비스에서 감지하거나 응용 서비스가 재시작할 때 현재 Log Size의 값을 ETCD에서 읽어온다.
(2) 응용 서비스 내에서 현재 동기화의 작업이 수행되고 있지 않고 응용 서비스의 Latest Processed Index의 값이 가져온 Log size의 값보다 작다면 (3)~(5) 과정을 수행한다.
(3) 인덱스 범위가 (Latest Processed Index의 값+1, Log Size의 값)에 해당하는 Request Log Entry의 값을 ETCD에서 가져온다.
(4) 응용 서비스는 인덱스가 가장 낮은 Request Log Entry부터 사용해 상태를 변경하거나 요청에 대한 처리 및 응답을 수행한다.
(5) 하나의 Request Log Entry에 관해 처리가 끝날 때 응용 서비스의 Latest Processed Index의 값을 해당 Request Log Entry의 인덱스로 업데이트한다.

2. Service registry

서비스 레지스트리는 서비스 복제 기법이 적용된 이더리움 블록체인의 응용 서비스를 지진, 테러와 같은 재난이나 다양한 해킹 같은 상황에서도 견고한 실행을 지원하기 위하여 이더리움 스마트 컨트랙트로 설계한다. 그림 2는 이더리움 스마트 컨트랙트로 설계된 서비스 레지스트리의 구조를 나타낸다.

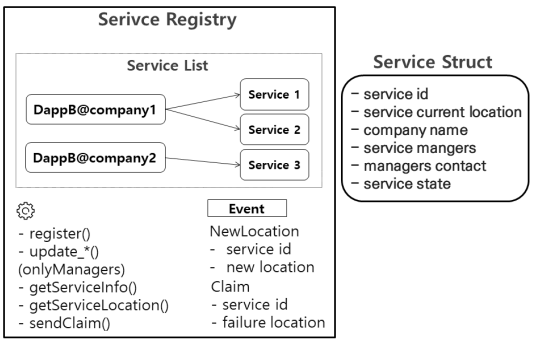

Fig. 2. Structure of service registry

그림 2에서 서비스 레지스트리의 register()는 서비스 관리자가 중요한 서비스 정보를 서비스 레지스트리 내에 저장하는 기능이다. 이때 저장되는 서비스 정보는 그림 2에 Serivce Struct의 형태로 저장되며 서비스의 ID, 접근 위치, 제공자와 같은 정보로 구성된다. 이러한 정보는 고유한 서비스 ID과 키-값 형태로 맵핑되어 서비스 레지스트리 내에서 서비스들의 정보를 담는 서비스 리스트에 저장된다. 또한 서비스 레지스트리는 이미 저장된 서비스의 정보를 변경하는 기능인 Update_*() 함수들이 있다. 이 함수들은 실행되기 전에 서비스 관리자인지 검증하는 onlyManagers() 함수 변경자가 선행으로 실행되어 Update_*() 기능들은 서비스 레지스트리에 서비스 정보를 등록한 서비스 관리자만이 사용할 수 있다. 그림 3은 서비스 레지스트리를 활용한 서비스 복제 기법을 나타낸다.

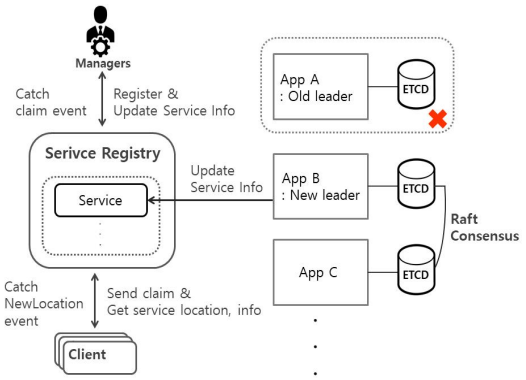

Fig. 3. Method of replicating services using service registry

서비스 복제 기법이 적용된 블록체인 응용서비스의 사용자는 이더리움 블록체인의 서비스 레지스트리를 이용해 재난 상태에서도 서비스의 접근 위치를 확인할 수 있다. 그림 3과 같이 기존 리더 응용 서비스가 장애나 결함과 같은 이유로 중지됐을 때 ETCD에 의해 실행되고 있는 응용 서비스들 사이에서 사용자에게 서비스를 제공할 새로운 리더가 선출된다. 새로운 리더 응용 서비스는 서비스 레지스트리 내에 등록된 해당 서비스의 접근 위치를 자신의 위치로 수정한다.

서비스의 접근 위치에 대한 수정이 완료되면 서비스 레지스트리는 서비스 접근 위치가 변경된 서비스 ID 및 위치를 알려주는 NewLocation 이벤트를 서비스 사용자에게 전달한다. 이 이벤트를 통해서 서비스 사용자는 새로운 서비스 접근 위치를 확인할 수 있다. 또한 이벤트를 받지 못한 서비스 사용자는 서비스 레지스트리에 getServiceLocation() 기능을 이용해 새로운 서비스 접근 위치를 얻어 낼 수 있다.

블록체인 응용 서비스 사용자가 모든 응용 서비스의 실패나 현재 서비스 접근 위치에서 서비스를 받지 못할 때 그림 3에 있는 서비스 관리자에게 복구 요청을 보내는 sendClaim() 기능을 통해 실패한 서비스에 대한 복구 요청을 할 수 있다. 서비스 레지스트리에서 sendClaim() 기능이 실행되면 서비스 ID과 현재 실패한 서비스 접근 위치 정보가 담긴 Claim 이벤트가 발생한다. 서비스 레지스트리에서 이 이벤트를 구독한 서비스 관리자는 발생한 Claim 이벤트를 통해 실패한 응용 서비스에 대한 정확한 위치를 신속하게 파악할 수 있다.

IV. Sample service for testing

앞 장에서 제안한 기법을 적용한 시범적인 응용 서비스를 개발하고 테스트를 통해 그 유용성을 확인한다. 개발할 응용 서비스는 Service Registry DashBoard로 서비스 레지스트리 컨트랙트에 서비스의 정보를 등록, 수정 및 보기 기능에 대한 유저 인터페이스를 제공하며 서비스 사용자가 발생시킨 서비스 복구 요청 이벤트도 보관하여 관리자에게 보여주는 블록체인 웹 응용 서비스이다. 그림 4는 개발된 Service Registry DashBoard의 유저인터페이스를 나타낸다.

1. Sample service

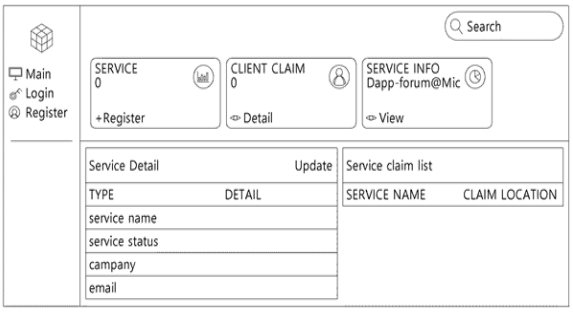

Fig. 4. UI of developed Service Registry Dashboard

Service Registry DashBoard의 구현은 웹 서버 프레임 워크인 Express를 기반으로 구현하며 서비스 복제 기법의 사용자 요청 저장과 동기화 작업을 스레드 단위로 개발한다. 표1은 Service Registry DashBoard 구현에서 서비스 복제 기법이 적용된 부분이다.

Table 1. Source code for replicating service

```
Main Thread:
 1: etcdC.watchStateChange(srClient)
 2: app.isSync = false
 3: app.stateMachine = new stateMachine()
 4: ...
Store request Thread:
 5: storeRequest = new Thread(()=>{
 6:   receiver(function(request, response){
 7:     request.set('LeaderID', NODE_ID)
 8:     connectPool.set(nodeID, response)
 9:     reqLog = etcdC.reqToLog(request)
10:     index = etcdC.get("LogSize")+1
11:     etcdC.put('Log/'+index , reqLog)
12:     etcdC.update("LogSize", index)
13:  })
14: })
Sync Thread:
15: sync = new Thread(()=>{
16:   lpi = etcdC.get('LPI/'+ NODE_ID)
17:   curLogSize = etcdC.get('LogSize')
18:   app.stateMachine.sync(lpi,curLogSize)
19:     etcdC.watchEvent('LogSize', function(event){
20:       app.curLogSize = event.value
21:       app.stateMachine.sync(lpi,curLogSize)
22:     })
23: })
```

1라인은 ETCD 클라이언트 모듈인 etcdC를 이용해 자신과 연결된 ETCD 상태가 리더인지를 지속적으로 체크한다. 자신과 연결된 ETCD의 멤버가 리더가 되면 서비스 레지스트리 클라이언트 모듈인 srClient를 이용하여, 서비스 레지스트리 내에 저장된 서비스 접근 위치를 해당 응용 서비스의 위치로 수정한다. 3라인에서 사용자 요청에 따른 응용 서비스의 상태 변경을 해주는 StateMachine을 생성한다.

Store request Thread는 서비스 복제 기법에서 사용자 요청을 저장 작업을 수행하는 스레드이다. 6라인에 receiver() 함수를 통해 사용자 요청인 request와 사용자 요청 커넥션인 response를 받아온다. 7~12라인을 거쳐 리더 응용 서비스는 사용자 요청을 저장하게 된다. Sync Thread는 서비스 복제 기법에서 동기화 작업을 수행하는 스레드이다. 16~18라인은 응용 서비스가 시작할 때 Latest Processed Index와 Log Size의 값을 ETCD에서 가져오고 sync() 함수를 실행하여 저장된 사용자 요청에 대한 상태 변경을 수행한다. 19 라인은 watchEvent()를 실행 시켜 ETCD에 사용자 요청 저장에 의한 Log Size의 값이 업데이트될 때 발생하는 이벤트를 구독하여 동기화 작업이 수행될 수 있도록 한다.

2. Test

본 절에서는 복제 기법이 적용된 블록체인 응용 서비스인 Service Registry DashBoard가 네트워크 장애와 같은 재난 환경에서도 복제된 응용 서비스가 정확하게 동작하는지에 대해 검사한다. 그림 5는 테스트 환경을 나타낸다.

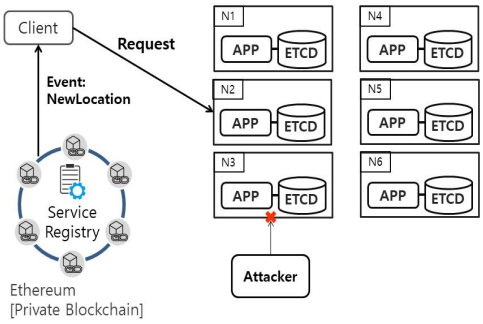

Fig. 5. Test environment

본 테스트는 6개의 독립적인 노드가 있는 분산 환경에서, 각 노드에 ETCD를 구성하는 멤버와 Service Registry Dashboard의 응용 서비스를 실행한다. 그림 5에서 Attacker는 특정 주기마다 응용 서비스가 실행되고 있는 노드에 일시적인 네트워크 장애를 일으키는 프로세스이다. 네트워크 장애 지속 시간은 이더리움 블록체인의 블록 생성 주기가 평균 10~15초이고 서비스 레지스트리에 새로운 정보가 기록되는 시간을 고려하여 20초로 설정한다.

본 테스트 환경에서 이더리움 블록체인은 6개 노드로 구성된 프라이빗 체인이며 이더리움 블록체인에 배포된 서비스 레지스트리는 클라이언트에게 NewLocation 이벤트를 전달하여 새로운 서비스 접근 위치를 제공한다. 개발된 Service Registry Dashboard의 테스트 과정은 다음과 같다.

(1) 클라이언트는 10분 동안 초당 2번의 사용자 요청을 보낸다. 이때 사용자 요청에는 임의의 난수가 포함되어 있다. 또한 Attacker도 클라이언트에서 사용자 요청을 보내는 동안 네트워크 장애를 특정 주기마다 발생시킨다.

(2) 리더 응용 서비스는 클라이언트로부터 요청을 받을 때마다 사용자 요청 저장 작업을 수행한다. 동기화 작업에서 복제된 각 응용 서비스는 사용자 요청에 있는 임의의 난수를 리스트에 저장한다. 리더 응용 서비스는 임의의 난수에 대한 저장이 완료되면 클라이언트에게 성공 응답을 보낸다.

(3) 클라이언트는 사용자 요청에 대해 성공 응답을 받으면 자신이 보낸 임의의 난수를 리스트 형태로 저장한다.

(4) 10분 동안 테스트가 완료되면 각 응용 서비스가 가지고 있는 리스트와 클라이언트가 가지고 있는 리스트가 완전히 일치하는지 검사하여 동기화 성공 여부를 확인한다.

Table 2. Test result

test1: Random Test test2: Leader Test			
Frequency of network failure	Failed requests (Out of 1200 requests)	Leader changes	synchroni-zation
60sec	test1: 8	test1: 1	success
	test2: 441	test2: 10	success
30sec	test1: 79	test1: 3	success
	test2: 564	test2: 20	success
25sec	test1: 137	test1: 6	success
	test2: 976	test2: 25	success

표 2의 테스트 결과는 랜덤 테스트와 리더 테스트가 있다. 랜덤 테스트는 임의의 노드를 선택해 네트워크 장애를 일으키는 테스트이다. 이 테스트에서 네트워크 장애에 회복한 응용 서비스들이 최신 상태를 동기화할 수 있는 지와 서비스를 제공하는 리더 응용 서비스가 실패하였을 때도 클라이언트에게 다시 서비스를 제공할 수 있는지를 확인한다. 표 2에 있는 랜덤 테스트의 결과에서 볼 수 있듯이 네트워크 장애에도 모든 응용 서비스의 상태가 성공적으로 동기화하였고 서비스를 제공하는 리더 응용 서비스의 실패에도 새로운 리더의 선출과 서비스 레지스트리를 통해 다시 서비스를 제공할 수 있음을 확인했다.

리더 테스트는 특정 주기마다 리더 응용 서비스가 실행되고 있는 노드에만 네트워크 장애를 일으키는 테스트이다. 이 테스트에서는 지속적인 리더의 실패에도 복제된 응용 서비스가 정확하게 상태를 동기화하는 지 확인한다. 표 2에 있는 리더 테스트의 결과에서 볼 수 있듯이 리더의 잦은 실패에도 상태에 대한 동기화는 성공적으로 달성됨이 확인되었다.

V. Conclusions

본 논문에서는 블록체인 응용 서비스의 견고한 실행을 지원하기 위하여 서비스 복제 기법과 서비스 레지스트리에 대하여 제안하였다. 이더리움 블록체인의 스마트 컨트랙트로 개발된 서비스 레지스트리를 통하여 재난 사태에도 서비스 정보를 관리, 서비스에 대한 유용한 정보 및 복구 요청 기능을 지원하도록 하였다. 이를 기반으로 장애나 결함에 의한 서비스 실패에도 서비스를 제공할 수 있는 새로운 노드에 대한 정보를 복원하여 서비스가 지속적으로 제공되도록 하였다. 또한 이를 위한 ETCD 키-값 분산 저장소를 기반으로 개발된 기법을 응용 서비스에 적용하여 보다 견고한 실행을 지원하였다. 더불어 시범 응용 서비스인 Service Registry DashBoard를 개발하여 서비스 관

리자가 서비스 레지스트리를 손쉽게 사용할 수 하였으며, 개발된 시범 응용 서비스의 정확성 및 견고성에 대한 테스트를 수행하여 해당 기법의 유용성을 확인하였다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(No. 2019R1I1A3A01052970)

REFERENCES

[1] T. Aste, et al., "Blockchain Technologies: The Foreseeable Impact on Society and Industry", Computer, Vol. 50, No. 9, pp. 18-28, Sep 2017. DOI: 10.1109/mc.2017.3571064

[2] S. Ølnes, A. Jansen, "Blockchain Technology as s Support Infrastructure in e-Government", Proceedings of the international conference on electronic government, pp. 215-227, Sep 2017. DOI:10.1007/978-3-319-64677-0_18

[3] H. Hou, "The application of blockchain technology in E-government in China", 2017 26th International Conference on Computer Communication and Networks(ICCCN), pp. 1-4, Jul 2017. DOI:10.1109/icccn.2017.8038519

[4] S. Ølnes, "Beyond Bitcoin Enabling Smart Government Using Blockchain" Electronic Government, pp. 253-264, Sep 2016. DOI:10.1007/978-3-319-44421-5_20

[5] P. Treleaven, R. G. Brown, D. Yang, "Blockchain technology in finance", Computer, Vol. 50, No. 9, pp. 14-17, Sep 2017. DOI:10.1109/MC.2017.3571047

[6] F. Casino, et al., "A systematic literature review of blockchain-based applications: current status, classification and open issues", Telematics and Informatics, Vol. 12, No. 8, pp. 55-81, Mar 2019. DOI:10.1016/j.tele.2018.11.006

[7] J. Sidhu, "Syscoin: A peer-to-peer electronic cash system with blockchain-based services for E-business", 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1-6, July 2017. DOI:10.1109/icccn.2017.8038518

[8] S. Huckle, et al., "Internet of Things blockchain and shared economy applications", Procedia Comput. Sci, Vol. 98, pp. 461-466, Sep 2016. DOI:10.1016/j.procs.2016.09.074

[9] M. Mettler, "Blockchain technology in healthcare: The revolution starts here", 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), pp. 1-3, Sep 2016. DOI:10.1109/healthcom.2016.7749510

[10] D. Ongaro, J. Ousterhout, "In search of an understandable consensus algorithms", In USENIX Annual Technical Conference (ATC), pp. 305-320, June 2014.

[11] P. McCorry, et al., "A smart contract for boardroom voting with maximum voter privacy", International Conference on Financial Cryptography and Data Security, pp. 357-375, April 2017. DOI:doi:10.1007/978-3-319-70972-7_20

[12] EOS, *https://eos.io/*

[13] M. Wohrer, U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity", Blockchain Oriented Software Engineering (IWBOSE) 2018 International Workshop on, pp. 2-8, Mar 2018. DOI:10.1109/iwbose.2018.8327565

[14] G. Destefanis, et al., "Smart contracts vulnerabilities: a call for blockchain software engineering?", Blockchain Oriented Software Engineering (IWBOSE) 2018 International Workshop on, pp. 19-25, Mar 2018. DOI:10.1109/iwbose.2018.8327567

[15] D. Huang, X. Ma, S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains", IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 50, No. 1, pp 172-181, January 2020. DOI:10.1109/tsmc.2019.2895471

[16] D. Woos, et al., "Planning for change in a formal verification of the raft consensus protocol", Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, pp. 154-165, Jan 2016. DOI:10.1145/2854065.2854081

[17] Hyperledger, *https://www.hyperledger.org/*

[18] Agero, *https://www.aergo.io/*

[19] ETCD, *https://etcd.io/*

[20] Brewer, Eric "Kubernetes and the New Cloud", Proceedings of the 2018 International Conference on Management of Data, pp. 1, May 2018. DOI:10.1145/3183713.3183725

[21] A. Freeman, "Essential Docker for ASP.NET Core MVC", pp. 1-6, 2017

[22] L. Lamport, "Paxos Made Simple", IEEE Transactions on Dependable and Secure Computing, pp. 1-52, Jan 2001

[23] J. Z. Konczak, et al., "Recovery Algorithms for Paxos-based State Machine Replication", IEEE Transactions on Dependable and Secure Computing, pp. 1-1, July 2019. DOI:10.1109/tdsc.2019.2926723

Authors

Min-Ho Kwon received the B.S/B.A degrees in IT convergence/Economics from University of Ulsan, Korea, in 2020. He is currently an M.S student in Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan. He is interested in blockchain technology, distributed computing, and cloud computing.

Myung-Joon Lee received the B.S. degree in Mathematics from Seoul National University in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST in 1982 and 1991, respectively. Dr. Lee joined the faculty of the Department of Computer Science at University of Ulsan, Ulsan, Korea, in 1982. He is currently a Professor in the School of IT Convergence, University of Ulsan. He is interested in blockchain technology, distributed computing, and mobile/cloud service.