Smart Media Journal / Vol.12, No.3 / ISSN:2287-1322
https://dx.doi.org/10.30693/SMJ.2023.12.3.60

Hyperledger Indy의 가용성을 위한 모니터링 시스템 구현

(Implementation of monitoring system for availability of Hyperledger Indy)

최규현*, 김근형**
(Gyu Hyun Choi, Geun Hyung Kim)

요 약

Hyperledger Indy는 탈중앙화 신원 증명 기술인 DID를 구현한 오픈소스다. Hyperledger Indy는 RBFT 합의 알고리즘을 사용하며 풀에 일정 수 이상의 문제 노드로 합의 부족이 발생하면 데이터 추가가 되질 않는다. 이는 노드를 추가하는 것으로 미연에 예방할 수 있으며 자동으로 동작시키기 위해 노드 모니터링 시스템을 구현하였다. 노드 모니터링 시스템은 풀의 상태를 지속해서 확인하며 특정수 이상의 문제 노드가 발생하면 자동으로 노드를 추가해 합의 문제가 발생하는 것을 방지한다. 이는 Hyperledger Indy의 가용성을 높여 줄 수 있으며 합의 알고리즘을 사용하는 다양한 블록체인 서비스에서 참고할 수 있는 연구이다.

■ 중심어 : 분산 식별자 ; 합의 알고리즘 ; 하이퍼레저 인디 프로젝트 ; 모니터링 시스템

Abstract

Hyperledger Indy is an open-source implementation of DID, a decentralized identity verification technology. Hyperledger Indy uses the RBFT consensus algorithm, and if there is a lack of consensus with more than a certain number of problem nodes in the pool, data is not added. This problem can be prevented in advance by adding a node, and a node monitoring system was implemented to operate automatically. The node monitoring system continuously checks the status of the pool and automatically adds nodes when there are more than a certain number of problematic nodes to prevent consensus problems from occurring. This proposed method can increase the availability of Hyperledger Indy and is a study that can be referenced in various blockchain services that use consensus algorithms.

■ keywords : Decentralized Identifier ; Consensus Algorithm ; Hyperleder Indy project ; Monitoring System

Ⅰ. 서 론

가상화폐를 통해 블록체인 기술이 조명받기 시작하면서 블록체인을 활용한 다양한 기술들이 나오고 있다. 그중 신원인증에 블록체인을 접목한 기술이 바로 Decentralized Identifier(이하 DID)이다. DID는 제 3자를 거치지 않고 자신을 인증할 수 있으며 모바일 신분증과 같은 다양한 신원인증 기술에 접목해 탈중앙화 신원인증 환경을 구성할 수 있다[1]. 이런 DID 기술을 개발하기 위해 다양한 프로젝트들이 만들어졌으며 그중 Hyperledger Foundation에서 진행한 Hyperledger Indy 프로젝트 또한 DID 기술을 구현하기 위해 만들어졌다.

Hyperledger Indy를 사용해 탈중앙 신원인증 기술을 모바일 신분증과 같은 다양한 기술들에 접목해 구현해 보았으며 DID를 사용한 다른 기술들 또한 분석하였다[1,2]. 이 과정에서 Hyperledger Indy가 가용성에 문제가 있는 것을 확인하였다. 노드 추가 테스트를 하던 도중 갑자기 블록체인에 데이터가 추가되지 않는 현상이 발생했다. 이는 블록체인에 사용되는 합의 알고리즘

* 학생회원, 동의대학교 게임공학과
** 정회원, 동의대학교 게임공학과
이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2021R1F1A1047573).

접수일자 : 2023년 02월 17일
수정일자 : 1차 2023년 03월 14일
게재확정일 : 2023년 03월 21일
교신저자 : 최규현, 김근형 e-mail : giry8647@gmail.com

때문이며 문제가 있는 노드들에 의해 합의가 원활하게 이루어지지 않아 생기는 현상이다. 이를 해결하기 위해선 문제 노드 수에 따라 노드를 추가해주는 기능이 필요하다[3].

본 논문의 구조는 다음과 같다. II 장에서 블록체인, 분산 네트워크, Decentralized Id, Hyperledger Indy 프로젝트와 같은 관련 지식을 살펴보고 III 장에서 제안한 노드 모니터링 시스템 및 테스트 결과를 기술한다. 최종적으로 IV 장에서 결론을 맺는다.

Ⅱ. 관련 지식

1. 블록체인과 분산 네트워크

블록체인이란 데이터인 블록들을 체인으로 묶어 데이터의 수정 및 삭제 여부를 쉽게 판별할 수 있는 기술이다. 블록체인에서 블록은 데이터를 뜻하며 체인은 이전 블록들의 데이터와 현재 블록의 데이터를 합친 해시값을 뜻한다. 블록체인에 데이터를 추가할 때 데이터를 블록에 담은 뒤, 체인으로 이전의 데이터들과 현재의 데이터를 엮어 저장한다. 이러한 방식은 이전 데이터들의 변형 및 삭제를 알기 쉽게 확인할 수 있다.

분산 네트워크란 데이터를 분산시켜 관리하는 기술이며 블록체인 기술의 핵심이다. 처음 블록체인을 만들면 일정 수 이상의 참가자들과 함께 네트워크를 만들게 되며 블록체인을 모두 공유한다. 각각의 참여자들은 블록체인의 데이터가 변조되었는지를 확인하며 검증한다. 데이터 추가는 참가자들이 추가할 데이터를 검증해 일정 수 이상의 합의를 통해 추가된다.

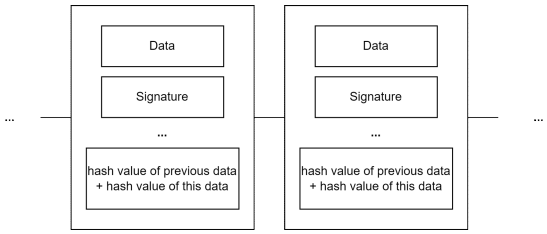

그림 1. 블록체인

블록체인은 분산 네트워크를 통해 데이터를 분산시켜 특정 기관에 데이터가 몰리는 것을 방지할 수 있으며 합의를 거쳐 데이터를 추가하므로 데이터의 안정성이 보장된다. 이는 데이터의 위변조를 막을 수 있어 큰 장점이다. 하지만 단점 또한 존재한다. 합의를 위해 일정 수 이상의 참가자가 네트워크에 존재해야 하며 데이터 추가에 합의가 필요하므로 속도가 느려질 수 있다. 또한 데이터가 분산되기 때문에 데이터가 노출될 수 있다.

2. Decentralized Id

DID는 자기주권 신원인증을 가능하게 만들어주며 블록체인을 사용해 탈중앙화된 환경에서 자신을 인증할 수 있다[4]. 사용자는 DID라는 분산 식별자를 사용하며 이는 사용자가 만들어 제어한다. 사용자가 DID를 만들 때 키 쌍이 함께 만들어지며 지갑에 저장된다. 지갑은 자신이 사용하는 다양한 기기에서 사용할 수 있으나 외부의 노출을 조심해야 한다. 지갑에는 기본적으로 DID와 키 쌍이 저장되며 증명 방법, 메타 데이터 등 추가적인 정보를 저장할 수 있다. DID 사용을 위해선 블록체인에 DID, 공개 키가 저장되어 있어야 하며 신원 증명에 필요한 증명 방법 또한 저장되어 있어야 한다. 블록체인에 저장한 각각의 데이터들은 조회 시 하나의 문서로 확인할 수 있으며 이를 DID Documents(이하 DID Doc)라고 한다.

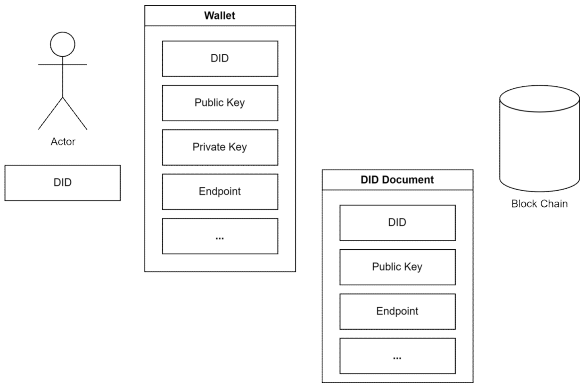

그림 2. DID 및 DID Doc

DID를 사용한 증명은 상대방에게 DID를 전달해주는 것으로 시작한다. 상대방은 받은 DID를 사용해 블록체인에서 DID Doc을 조회한다. DID Doc의 공개 키를 사용해 증명을 위한 메시지를 암호화하며 DID Doc의 증명 방법을 통해 메시지를 전달한다. DID의 소유주는 받은 암호화 메시지를 개인 키를 사용해 복호화하여 이를 상대방에게 전달한다. 이를 통해 DID Doc에 있는 공개키의 주인임을 인증할 수 있으며 DID의 소유주임을 증명하게 된다.

3. Hyperledger Indy 프로젝트

Hyperledger Indy는 Hyperledger Foundation에서 관리하는 오픈소스 프로젝트로 블록체인 또는 분산 원장에 기반한 DID를 제공한다. Hyperledger Indy는 프라이빗 블록체인을 통해 블록체인의 참여자를 제한하는 것으로 정보의 유출을 막거나 합의 속도를 높일 수 있다. Hyperledger Indy는 DID 기술을 구현하기 위해 분산 원장 기술을 가진 'indy-node'와 이러한 분산 원장의 Application Programming Interface(이하 API)를 제공하는 'indy-sdk'로 나뉜다[5].

가. indy-node

'indy-node'는 블록체인 기술의 핵심인 분산 원장 기술이 구현되어 있다. 'indy-node'는 분산 원장을 통해 데이터의 탈중앙화를 구현하고 합의 과정을 통해 데이터가 추가된다. 'indy-node'에선 각각의 사용자들을 노드라고 하며 노드들로 이루어진 하나의 네트워크를 풀이라고 한다. 각각의 노드들은 실행 시 풀의 정보가 담긴 풀 제네시스 파일이 필요하며 실행 이후 풀 제네시스 파일에 담긴 노드에 통신을 보내 기존 원장 정보를 요청한다. 실행 이후 바로 풀에 참가하진 않으며 풀에 노드 추가 요청을 보내 통과하면 정식으로 풀에 참여하여 원장 정보 요청이 가능하다.

'indy-node'에서 원장은 노드에서 각각의 정보를 저장하는 저장소 개념으로 보통 트랜잭션의 로그들을 저장한다. 노드에 'indy-sdk'를 통한 트랜잭션이 들어올 때 원장에 이를 저장하며 요청할 수 있다. 'indy-node'의 주된 원장은 4가지가 있으며 각각 Pool 원장, Domain 원장, Config 원장, Audit 원장이 있다. Pool 원장은 풀에 소속된 노드들의 정보가 담긴 원장이며 노드들에 대한 트랜잭션이 저장된다. Domain 원장은 주요 애플리케이션 기능이 들어가는 원장으로 DID 및 스키마 같은 DID와 관련된 트랜잭션들이 저장된다. Config 원장은 풀 설정과 관련된 정보가 저장되는 원장으로 풀 공통의 매개변수, 풀 업그레이드 정보 등이 담겨있다. Audit 원장은 다른 노드들의 연결 및 상태를 저장하는 원장으로 새 노드 및 노드 동기화 등에 사용된다[6]. 풀에 참여한 노드들은 서로 지속해서 통신하며 원장 상태를 검증한다.

나. indy-sdk

'indy-sdk'는 'indy-node'를 통해 만들어진 풀을 이용하기 위한 API를 제공해주며 이를 이용해 원장 쓰기 및 풀 상태 확인 등 풀에 다양한 요청을 할 수 있다. 그 외에 지갑 및 DID 생성 등 DID 사용을 위한 기본적인 기능들 또한 제공한다. 'indy-sdk'는 풀에 소속된 노드들의 정보가 담긴 풀 제네시스 파일을 통해 풀과 연결할 수

있으며 풀 제네시스 파일에 있는 랜덤한 노드에 요청을 보내 답을 받는다. 노드에 특정 요청을 보내기 위해선 트랜잭션 작성이 필요하며 요청에 따라 특정 권한이 필요하다. Hyperledger Indy는 프라이빗 블록체인이기 때문에 일정 이상의 권한을 소유하지 않으면 원장에 데이터를 쓰는 것이 불가능하다. 이는 권한을 가진 관리자에게 요청하여 권한을 부여할 DID를 풀에 저장한 뒤, 특정 권한을 부여해 주는 것으로 해결할 수 있다. 권한은 TUSTEE, STEWARD, ENDORSER, NETWOEK_MONITOR가 있으며 TRUSTEE의 권한이 제일 높다.

다. RBFT 합의 알고리즘

Hyperledger Indy 또한 블록체인을 사용한 분산 네트워크 기술이 들어가 있으며 합의 알고리즘이 존재한다. Hyperledger Indy는 Redundant Byzantine Fault Tolerance (이하 RBFT) 합의 알고리즘을 사용한다[7]. RBFT 합의 알고리즘은 일부의 문제 노드들로 인해 최종적인 합의가 이루어지지 않는 것을 방지하기 위해 일정 비율 이상의 노드가 합의하면 최종적으로 데이터를 추가한다. RBFT 합의 알고리즘은 다음과 같은 수식을 가진다[8].

$$3f + 1 <= N \quad (1)$$

수식(1)에서 f는 합의하지 않거나 문제가 있는 노드 수이며 N은 풀에 소속되어 있는 총 노드 수다. 즉, f만큼의 참여자가 합의를 진행하지 않더라도 총참여자 수가 3f + 1보다 많으면 합의가 이루어졌다고 판단하여 최종적인 데이터 추가가 이루어진다. 예를 들어 총 4개의 노드가 있는 풀에 1개의 노드가 모종의 이유로 합의를 하지 않더라도 남은 3개의 노드의 합의가 있다면 최종적으로 데이터 추가가 이루어진다. 만약 2개 노드가 모종의 이유로 합의를 하지 않는다면 남은 2개 노드의 합의를 얻더라도 수식을 만족하지 않기 때문에 데이터 추가가 이루어지지 않는다. 위 수식(1)을 통해 문제 노드 수인 f가 증가할 때 총 노드 수인 N은 3의 배수로 증가해야 수식을 만족한다는 것을 알 수 있다.

Ⅲ. 노드 모니터링 시스템

1. 노드 모니터링 시스템 동작

Hyperledger Indy는 RBFT 합의 알고리즘에 의해 일부의 문제 노드가 합의를 하지 않더라도 수식(1)을 만족한다면 최종 합의가 이루어진다. 그러나 문제 노드 수가 많아 수식(1)을 만족하지 않으면 합의가 이루어지지 않는다. 이는 총 노드 수를 증가시켜 수식(1)을 만족시키는 것으로 해결할 수 있다. 노드 모니터링 시스템은 노드를 미리 추가하는 것으로 총 노드 수를 증가시켜 문제 노드 수 증가로 인한 합의 문제를 미연에 방지한다. 또한 문제 노드 수 증가에 맞춰 노드를 추가하는 것으로 이후에 생길 합의 문제 발생을 예방한다.

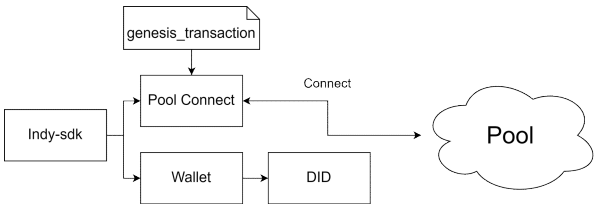

그림 3. DID 생성 및 풀 연결

그림 3은 풀 사용을 위한 기본적인 과정을 표현한 그림이며 처음 실행 시의 동작이다. DID 사용을 위해 기본적인 지갑을 생성하며 DID 생성을 진행한다. Hyperledger Indy는 프라이빗 블록체인이기 때문에 미리 인증된 DID가 필요하며 이는 시드 값을 사용해 만든다. 이후 'pool_transactions_genesis' 파일을 통해 풀에 연결한다. 외부 컴퓨터에 추가 노드를 생성하기 위한 정보를 가져오며 외부 컴퓨터의 IP 주소, 포트 번호, 실

행 가능한 최대 노드 수, 현재 가동 중인 노드 수, 노드 정보 경로가 있다. 본 논문은 외부 컴퓨터 사용을 위해 ssh(secure shell) 및 scp(secure copy)를 사용하였다. 노드 모니터링 시스템은 외부 컴퓨터의 정보를 통해 연결을 한 뒤, 현재 모니터링하고 있는 풀에 노드 정보를 요청한다.

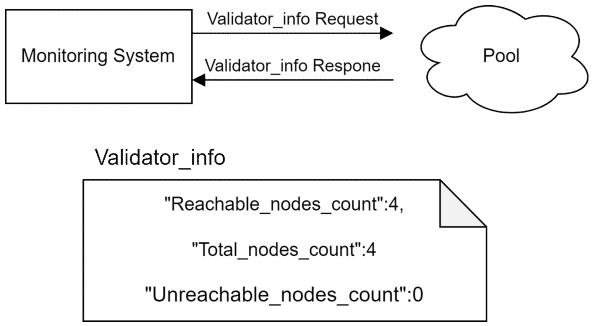

그림 4. Validator_info

그림 4는 풀에 노드 정보를 요청하는 과정이다. 노드 정보는 'Validator_info'라고 부르며, 'indy-sdk'의 API를 통해 요청할 수 있다. 'Validator_info'에는 풀에 소속된 노드들의 정보를 확인할 수 있다[9].

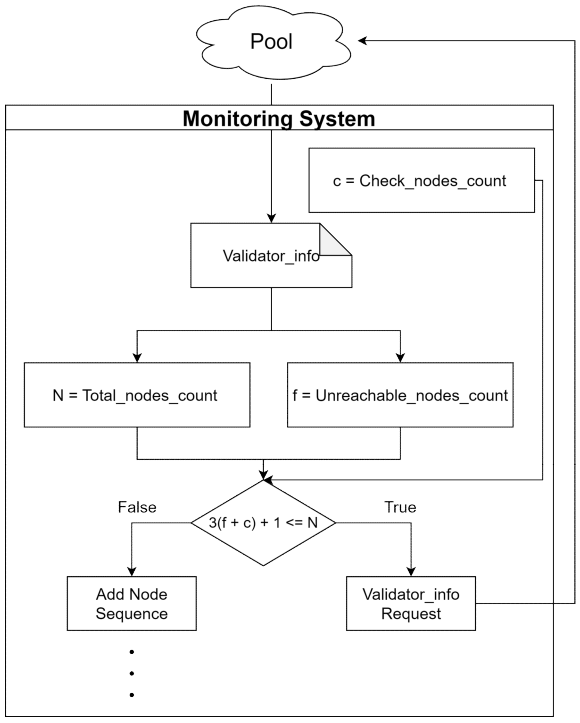

그림 5. 풀 상태 검증

노드 모니터링 시스템은 'Validator_info'에서 '총 노드 수'(그림5의 'Total_nodes_count')와 '문제 노드 수'(그림5의 'Unreachable_nodes_count')를 참조하여 풀 상태 검증에 사용한다. 풀 상태 검증 시 '문제 노드 수'에 특정 정수 값(그림5의 Check_nodes_count)을 더하며 수식(2)를 사용한다.

$$3(f + c) + 1 <= N \quad (2)$$

수식(2)은 수식(1)의 문제 노드 수에 특정 정수 값을 더한 수식이다. 수식(2)을 통해 검증하면 수식(2)을 만족시키기 위한 총 노드 수가 기존의 수식(1)보다 늘어나 일정 수 이상의 문제 노드 수 발생으로 인한 합의 문제를 미연에 방지할 수 있다. 풀 검증 과정은 그림5와 같다. 조건을 만족하면 풀 상태 검증을 반복하며 만족하지 못하면 노드 추가를 진행한다.

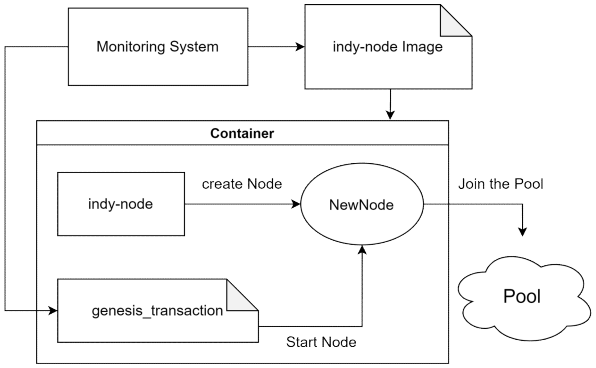

그림 6. 노드 생성 및 실행

모니터링 시스템에선 앞선 풀 상태 검증을 통해 문제 노드 수가 많아 풀이 위험하다고 판단되면 노드 추가를 진행하며 노드는 총 3개를 추가한다. RBFT 합의 알고리즘의 수식(1)에 따라 문제 노드 수인 f가 증가할 때 총 노드 수인 N은 3의 배수로 증가해야 수식을 만족할 수 있으므로 3개의 노드 추가가 필요하다. 또한 이전 연구에서 문제 노드 수 증가에 따라 노드를 추가하는

것으로 풀의 가용성이 증가하는지를 테스트하였다[3]. 문제 노드 수 1개에 3개의 노드를 추가했을 때 데이터 추가가 이루어지나 3개보다 적은 노드를 추가했을 때 데이터 추가가 이루어지지 않는 것을 확인했다. 이와 같은 이유로 3개의 노드를 추가한다. 그림 6은 노드 추가 과정이며 Docker를 이용해 진행한다. 이미지를 사용해 Container를 생성한다. 이미지는 노드 모니터링 시스템 시작 전에 미리 준비해두며 우분투 이미지를 기반으로 'indy-node' 설치와 기본적인 노드 설정을 작성한다. 노드 모니터링 시스템은 Container 내부에서 'indy-node'를 사용해 총 3개의 노드를 생성한다. 노드 모니터링 시스템에서 연결 중인 풀의 'pool_transactions_genesis' 파일을 받아와 만들어진 노드를 모두 실행한다. 노드 생성 및 실행이 모두 완료되면 만들어진 노드들의 정보를 노드 모니터링 시스템에 전달해준다. 노드 모니터링 시스템은 전달받은 정보들을 사용해 풀에 노드 추가 요청을 보낸다. 이후 풀 상태를 요청하여 노드 추가를 확인한 뒤, 문제 노드 수 증가에 맞춰 노드 추가를 반복하며 이후 풀 상태 검증을 반복한다.

2. 노드 모니터링 시스템 테스트

구현한 노드 모니터링 시스템을 실제 풀에 연결해 동작을 테스트하였다. 풀의 상태는 총 노드 수 10개, 문제 노드 수 0개, 정상 노드 수 10개이며 문제 노드 수를 3개까지 검증한다.

표 1. 노드 모니터링 시스템 테스트

순서	총 노드 수	동작 노드 수	문제 노드 수	추가 노드 수	체크 노드 수 + 문제 노드 수
1	10	10	0	0	3
2	10	9	1	0	4
3	13	12	1	3	4
4	13	10	3	3	6
5	19	16	3	9	6

표 1은 노드 모니터링 시스템의 테스트를 그림 5의 c(체크 노드 수)가 3으로 설정하고 수행한 결과이다. 풀을 구성하는 노드 중 일부를 의도적으로 문제 노드로 설정하고 이를 감지하여 노드를 추가하는지 확인하였다.

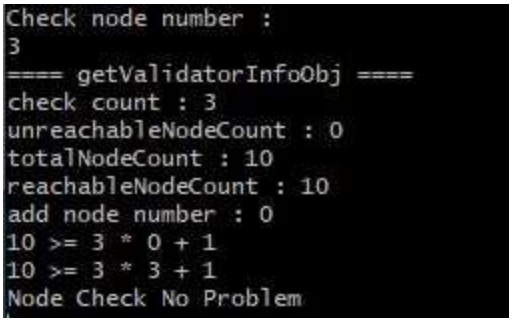

그림 7. 풀 상태 검증화면

그림 7은 풀 연결 이후 특정 정수 값을 입력받고 'Validator_info'를 요청해 풀 상태를 검증하는 화면이다. 현재까지 풀에 문제가 없으므로 정상적으로 작동한다.

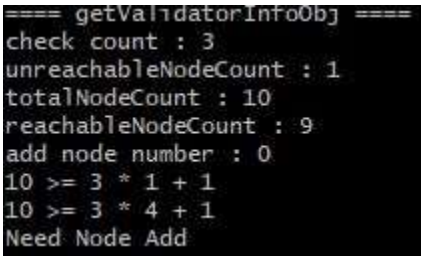

그림 8. 문제 노드 발생 이후 풀 상태 검증화면

노드 하나를 제거하면 그림 8과 같이 풀 상태 검증을 통해 문제 노드가 증가한 것을 확인할 수 있다. 또한 노드 모니터링 시스템의 노드 추가 조건에 만족하기 때문에 노드 추가가 이루어진다.

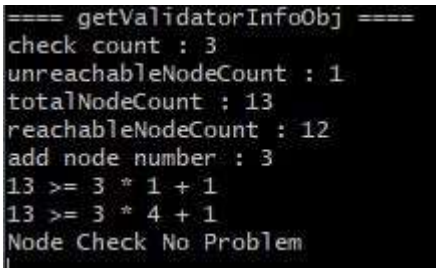

그림 9. 노드 추가 이후 풀 상태 검증화면

노드 추가가 이루어진 이후 다시 노드 정보를 요청해 풀 상태를 검증한다. 그림 9는 노드 추가 이후 풀 상태 검증화면이며 3개의 노드가 정상적으로 추가된 것을 확인할 수 있다.

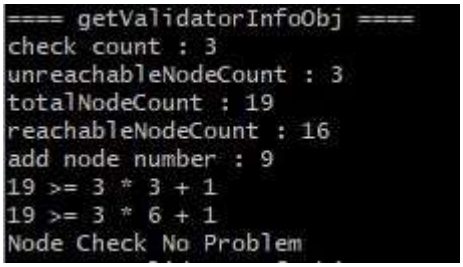

그림 10. 추가 테스트 이후 풀 상태 검증화면

이후 노드 두 개를 추가로 제거하면 노드 추가가 한번 이루어진 뒤 다시 풀 상태를 검증하여 노드 추가를 진행한다. 그림 10은 노드 추가를 2번 진행한 뒤 풀 상태 검증화면이며 노드 추가 이후 풀이 정상적으로 동작하는 것을 확인할 수 있다. 테스트를 통해 노드 모니터링 시스템이 합의 부족 문제를 해결할 수 있다는 것을 확인하였으며. 최종적으로 Hyperledger Indy의 가용성이 증가했다.

3. 노드 모니터링 시스템의 한계

테스트를 통해 노드 모니터링 시스템을 풀에 연결하는 것으로 가용성을 높일 수 있다는 것을 확인하였다. 하지만 노드 모니터링 시스템에는 한계가 있다. 풀의 노드 상황이 수식(1)을 만족하지 않으면 합의 알고리즘이 동작하지 않아 데이터 추가가 일어나지 않으며 이는 노드 추가 데이터 또한 같다. 결과적으로 노드 추가 작업을 진행하는 노드 모니터링 시스템이 동작하지 않는다. 이는 처음 풀 실행 시 수식(1)의 총 노드 수(N)를 늘려 문제 노드 수의 증가로 인한 합의 문제를 방지하는 방법이 있다.

Ⅳ. 결 론

Hyperledger Indy는 충분한 노드 수를 가지고 있지 않거나 문제 노드 수가 많을 때 데이터 추가가 일어나지 않는 문제를 가지고 있으며 이는 서비스 제공 관점에서 큰 문제이다. 이를 해결하기 위해 이전에 진행하였던 연구[3]를 바탕으로 노드 모니터링 시스템을 구현하였으며 실제 Hyperledger Indy의 가용성이 증가하는 것을 확인하였다. 본 연구 결과는 향후 Hyperledger Indy를 사용한 서비스를 제공할 때 유용할 것이며 합의 알고리즘을 사용하는 다양한 기술들에도 적용할 수 있다. 이후 추가적인 연구를 통해 노드 모니터링 시스템을 보완할 예정이다.

REFERENCES

[1] 최규현, 김근형, "자기주권 신원 생태계를 위한 신뢰할 수 있는 통신 방법," *정보처리학회논문지 컴퓨터 및 통신시스템,* vol. 11, no. 3, 통권 114호 pp. 91-98, 2022년

[2] 최규현, 김근형, "DIDComm 메시지의 암호화 및 서명 알고리즘," *한국멀티미디어학회 추계학술발표대회 논문집,* 25.2 (2022): 122-125

[3] 최규현, 김근형, "Hyperledger Indy 프로젝트 프로젝트 네트워크 가용성 향상을 위한 모니터링 시스템 설계," *한국정보처리학회 학술대회논문집,* 29.2 (2022): 252-254

[4] M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele and C. Allen, “Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations,” W3C PR, Aug. 2021.

[5] Tracy Kuhrt, Hyperledger Indy, January 3. 2019. https://wiki.hyperledger.org/display/Indy/Hyperledger+Indy (accessed April, 23, 2022).

[6] Storage components(2018). indy-plenum. December 19. 2021. https://github.com/hyperledger/indy-plenum/blob/main/docs/source/storage.md (accessed July 11, 2022).

[7] Sergey Khoroshavin, indy-node, “Indy Node troubleshooting guide”, March 10. 2020. https://github.com/hyperledger/indy-node/blob/main/docs/source/troubleshooting.md (accessed June 30, 2022).
[8] P. -L. Aublin, S. B. Mokhtar and V. Quéma, "RBFT: Redundant Byzantine Fault Tolerance," *2013 IEEE 33rd International Conference on Distributed Computing Systems 2013,* pp. 297-306, doi: 10.1109/ICDCS.2013.53.
[9] Alexander Shcherbakov. indy-node. “Extension of validator_info”, May 24. 2018. https://github.com/hyperledger/indy-node/blob/main/design/validator_info.md (accessed September 5, 2022).

저 자 소 개

최규현(학생회원)
2016년 동의대학교 게임애니메이션전공 재학
<주관심분야 : 블록체인, DID ,데이터 주권>

김근형(정회원)
1986년 서강대학교 대학교 (공학사)
1988년 서강대학교 대학원 (공학석사)
2005년 포항공과대학교 대학원 (공학박사)
2007년 ~ 현재 동의대학교 게임공학전공 교수
<주관심분야 : 탈중앙웹, Web3.0, 인공지능, 설명가능 인공지능, 블록체인, 자기주권 데이터>