

K-P2PLab: P2P 네트워크 토폴로지 분석을 위한 테스트베드 및 분석 플랫폼 개발

이 성 욱*, 주 흥 택°

K-P2PLab: Development Testbed and Analysis Platform for P2P Network Topology Analysis

Sung-wook Lee*, Hong-taek Ju°

요 약

본 논문은 다수의 노드가 연결된 P2P 네트워크를 생성하고 실행하는 P2P 네트워크 테스트베드를 만들고 만들어진 P2P 네트워크의 토폴로지 분석이 가능한 플랫폼 개발 결과를 제시한다. P2P 네트워크의 토폴로지는 노드들이 연결되어 있는 모양으로 P2P 네트워크의 성능에 영향을 미치는 중요한 요소이나 모든 노드의 연결 정보를 수집해야 토폴로지를 알 수 있어 실제 네트워크에서는 파악하기 매우 어렵다. 따라서 테스트베드를 구축하여 실시간으로 토폴로지의 변화나 특성을 분석하는 플랫폼이 필요하다. 우리는 공개 소프트웨어 중에서 높은 기술 수준과 많이 활용되고 있는 libp2p와 가상화 기술인 도커(Docker)를 활용하여 P2P 네트워크를 만들고 실행하며 각 노드의 연결 정보를 실시간으로 수집하여 토폴로지를 분석할 수 있는 플랫폼을 개발하였다. 개발한 결과를 검증하기 위하여 P2P 네트워크를 만들고 토폴로지를 분석한 결과를 제시하였다.

Key Words : P2P, Blockchain, Topology

ABSTRACT

This paper creates a P2P network testbed that creates and executes a P2P network in which a number of nodes are connected and presents the results of the development of a platform that can analyze the topology of the created P2P network. The topology of a P2P network is an important factor that affects the performance of the P2P network in the form of connecting nodes, but it is very difficult to grasp in a real network because the topology can be known only when all nodes' connection information is collected. Therefore, a platform that analyzes topology changes or characteristics in real-time by building a testbed is needed. We developed a platform that can analyze the topology by creating and executing a P2P network by using a high technology level among public software, the widely used libp2p, and the virtualization technology Docker, and collecting connection information of each node in real time. To verify the developed results, a P2P network was created and the results of topology analysis were presented.

※ 이 논문은 2023년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 이공분야기초연구사업임. (No. NRF-2023R1A2C2006045).

◆ First Author : Keimyung University Department of Computer Engineering, limos1479@gmail.com

° Corresponding Author : Keimyung University Department of Computer Engineering, juht@kmu.ac.kr

논문번호 : KNOM2024-02-08, Received December 6, 2024; Revised December 17, 2024; Accepted December 24, 2024

1. 서론

P2P 구조는 전통적인 클라이언트-서버 구조와 대비되는 네트워크 구조로, P2P 구조에서는 네트워크의 모든 참여자가 동등한 역할과 기능을 수행하는 피어가 된다. 네트워크를 주도하는 별도의 서버 없이 피어 간에 자율적으로 데이터를 주고받는 형태로 네트워크를 구축하게 되고, 이러한 P2P는 블록체인의 기술의 기반이 된다.

블록체인은 Satoshi Nakamoto가 제안한 분산형 디지털 원장 기술로, 신뢰 가능한 데이터 기록 및 관리를 구현하는 탈중앙화 된 네트워크이다. 블록체인은 블록이라는 형태로 데이터를 저장하는데, 이는 암호학적 해시를 통해 이전 블록과 연결되어 P2P 네트워크상에 연속적이고 변경 불가능하게 저장된다. 블록체인의 분산형 디지털 원장을 구현하기 위해 체인에 기록될 모든 데이터는 네트워크에 속한 다른 참여자에게 전파된다. 블록체인은 그 종류에 따라 다양한 종류의 데이터를 네트워크상에서 수집하여 블록에 작성하고, 이것이 네트워크에 다시 전파되어 PoW, PoS 등의 합의 알고리즘을 통해 체인에 영구적으로 연결된다. 블록체인은 P2P 네트워크를 통해 체인에 기록될 블록을 공유하고 합의과정을 수행한다. 모든 참여자 즉, 노드(=피어, P2P 수준에서는 피어, 호스트 수준에서는 노드라는 단어를 사용)는 데이터(블록과 트랜잭션)를 P2P 네트워크를 통해 전파되어 공유되고 모두가 검증하여 올바른 데이터임을 보장받아 탈중앙화된 구조를 구현하고 있다[1-2].

블록체인은 P2P 네트워크를 활용한 응용으로 활용가치가 높아져서 현재 많은 노드가 참여하여 규모가 매우 커져서 수만 개의 노드가 P2P 네트워크에 참여하고 있다. 블록체인 P2P 네트워크가 커짐에 따라 여러가지 문제가 발생하고 있는데, 대표적인 잘 알려진 문제가 바로 블록체인의 3도논법(Trilemma Problem) 문제이다[3]. 이것은 탈중앙화, 보안, 확장성을 모두 만족하는 해결방법이 존재하지 않는다는 문제이다. 블록체인 네트워크의 규모가 커짐에 따라 전체 노드가 블록과 트랜잭션을 공유하고 합의 시간이 오래 걸려 처리하는 초당 처리하는 트랜잭션의 수가 감소하여 확장성이 떨어지고 확장성을 높이기 위해서 특정 노드가 중요한 역할을 수행하면 탈중앙성이 떨어지게 된다. 합의 과정을 단순화시켜서 보안을 감소시키면 탈중앙화와 확장성을

증가시킬 수는 있다.

블록체인의 3도논법 문제를 해결할 수 있는 방안 중 하나는 블록체인 P2P 네트워크의 성능을 높이는 것이다. P2P 성능이 높아지면 블록이나 트랜잭션 전파가 빨라지고 합의 과정이 짧은 시간에 수행되어 초당 처리 트랜잭션 수를 증가시킬 수 있다. 따라서 현재의 보안과 탈중앙성을 유지하면서 확장성을 증가시킬 수 있는 방법이다[4].

블록체인 상의 트랜잭션 및 블록 전파는 브로드캐스팅 형태의 통신방법으로 이루어지고 있으며, 따라서 브로드캐스팅의 성능을 개선함으로써 전파 지연을 감소시킬 수 있다. 브로드캐스팅의 성능을 개선하기 위해서는 다양한 전파 방법, 매개 변수 및 알고리즘의 수정과 실험을 통해 결과를 도출하고, 이를 분석하여 각 실험의 브로드캐스팅 성능을 측정해야 한다. 그러나 전파 지연과 같은 지표를 실제 동작 중인 블록체인 네트워크에서 측정하기는 이전 연구를 통해 매우 어렵다는 것을 이미 알고 있다[5-7]. 그러므로 브로드캐스팅을 실험하고 성능을 측정하기 위한 실제 P2P 네트워크와 같은 방식으로 동작하는 실험 환경 구축이 필요하다.

우리는 이더리움 등에 사용되는 대표적인 P2P 라이브러리인 Go-libp2p[8]와 Docker[9] Swarm 네트워크를 사용해 외부와 분리된 독립된 가상의 P2P 네트워크를 구축하는 P2P 테스트베드, K-P2PLab을 개발하였다. K-P2PLab에서는 다수의 P2P 노드를 만들고 서로 연결하며 메시지를 브로드캐스팅하게 할 수 있다. 각 노드는 도커 네트워크상에서 각 컨테이너마다 하나의 피어로 실행되며 실험 목적에 따라 다양한 형태의 P2P 네트워크를 구축할 수 있다. 또한 구축된 P2P 네트워크에서 다양한 형태의 브로드캐스팅 메시지 전달이 가능하도록 하였다.

K-P2PLab은 Prometheus[10] 및 Grafana[11]를 활용하여 P2P네트워크를 실시간으로 모니터링하고, 파일 형태의 로그 수집해서 P2P 네트워크의 토폴로지, 브로드캐스팅 메시지 전달 과정 및 성능을 분석할 수 있다. 로그 데이터 수집 및 분석도 도커 기술을 활용하여 네트워크의 크기 및 성능 모니터링 데이터의 수집량에 따라서 탄력적으로 모니터링 및 분석 성능을 조절할 수 있게 하였다. 또한 도커를 사용하여 클러스터링 된 서버 내에서 실험 환경의 조절을 용이하게 하였다. 또한 이후에 추가될 네트워크 트래픽 지연의 구현에서도 도커를 사용하면 간단하게 해당 기능을 구현할 수 있다.

위 방법은 실제 네트워크를 간단하게 모방하고

실험할 수 있는 이점이 존재하나, 현재로서는 현실에서의 네트워크 지연이나 여러 네트워크 공격 등을 완벽히 구현하지 못한다는 한계점 등이 있다. 그럼에도 P2P 가상 실험 환경의 구현에 있어, 실험 환경 속 각 피어들이 모두 현실의 P2P 네트워크상의 피어와 같은 모듈을 사용하여 동일한 방식으로 동작하기에, 위 실험 환경에서 브로드캐스팅의 성능과 네트워크의 메시지 라우팅 경로 등을 측정하여 실제 네트워크를 추정 가능하게 한다. 이를 통해 P2P 네트워크의 전파 지연을 개선하기 위해 어떤 방법이 가능할지 소규모 네트워크에서 빠르고 안전하게 미리 적용해보고 테스트해 볼 수 있다는 가치를 지닌다. 이렇게 실험된 아이디어가 실제 환경에 적용되면, 우리의 테스트베드가 정확하게 동작한다는 가정하에 테스트베드 상의 성능 개선과 유사한 성능 개선을 보일 가능성이 높고, 이는 실제 P2P 네트워크 전체를 수정하는 리스크의 발생 횟수를 줄이는 데에 기여할 수 있다.

우리가 개발한 실험 환경에서 P2P 네트워크가 올바르게 작동하고 이 P2P 네트워크의 성능을 충분하고 정확하게 모니터링하고 분석할 수 있음을 P2P 네트워크 토폴로지 분석과 노드의 차수 분석을 시각화하여 검증하였다. 그리고 우리는 K-P2PLab을 통해 현재 libp2p의 연결 방식의 복잡성과 개선의 필요성을 발견할 수 있었다.

2장에서는 우리와 비슷한 목적을 가지고 플랫폼

을 개발한 관련된 연구에 대해서 언급할 것이다. 3장에서는 K-P2PLab을 개발하기 위해 겪은 시행 착오와 서버 단위의 네트워크 관점의 아키텍처에 대해서 소개한다. 이어지는 4장에서는 K-P2PLab의 노드 단위 애플리케이션과 컨테이너의 구조에 대해서 소개하고, 5장에서는 우리가 개발한 K-P2PLab을 검증하고, 직접 활용하여 알게 된 점에 대해서 기술한다. 마지막으로 6장은 결론과 향후 연구에 대해서 언급하며 본 논문을 마친다.

II. 관련 연구

이전에도 Whiteblock[12], Testground[13]와 P2P 네트워크와 성능을 모니터링하기 위한 도구가 있었다. 기존의 도구들 역시 P2P 네트워크의 테스트 환경 생성 및 해당 환경 속 각 피어의 로그를 수집하는 것이 가능하였다. Whiteblock은 외부에서 복제한 별도의 libp2p 프로젝트를 사용하여 Gossipsub의 전파 성능을 90개 내외의 노드를 가진 단일 호스트에서 실험하는 내용이다. 전체 프로젝트가 하나의 도커 컨테이너 내에서 여러 개의 P2P 어플리케이션 프로세스가 통신하는 방법이라 실제 물리적 서버의 성능에 의해 노드의 수를 증가시키는 데에 한계가 분명하다. Testground는 이보다는 이후의 프로젝트이며, 설정 파일을 사용하여 여러 개의 도커 컨테이너를 생성하여 실행할 수 있다. 여러 개의 도커 컨테이너에 각 하나씩 P2P 어플리케이션이 생성되어

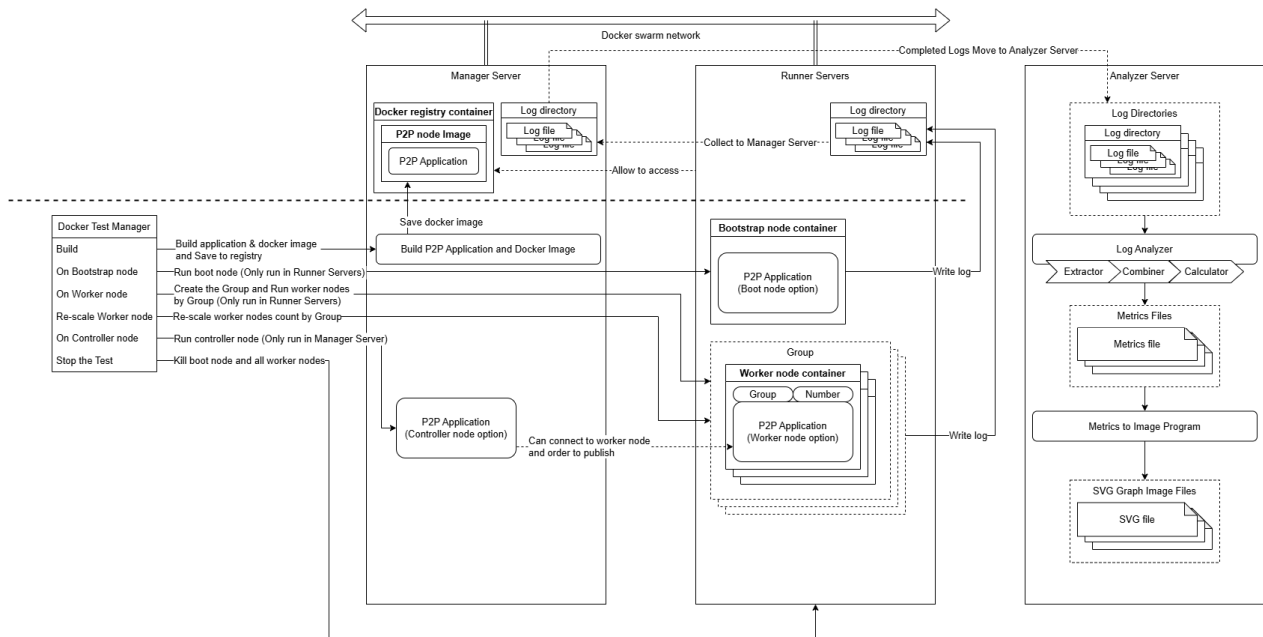


그림 1. K-P2PLab 구조도
Fig. 1. Architecture of K-P2PLab

P2P 네트워크를 형성하고 로그를 생성하고 전체 테스트베드 관리자가 이를 수집하고 분석하는 우리의 K-P2PLab과 유사한 구조를 가지고 있다. 하지만 local:docker 설정을 통한 단일 호스트의 배포만이 남아있고, 스웜 네트워크를 활용하여 클러스터링 여러 개의 된 서버를 활용하는 swarm:docker 설정은 우리가 확인한 바로는 과거에 지원이 종료된 상황이라 현재 상황으로는 제대로 사용하기 어려웠다.

Testground와 Whiteblock은 본 연구의 목적과 같은 목적으로 만들어진 도구들이고 사용한 기술도 동일하다. 도구의 사용법에 대한 설명이 잘 갖추어져 있고 공개 프로젝트로서 소스코드도 공개되어 있다. 그러나 이 두 도구는 기술적인 세부 내용을 자세히 기술한 기술 문서가 부족하다. 또한 Testground와 Whiteblock은 주요 핵심 소스 코드가 2년전 이후 더 이상 개선되거나 변경되지 않고 있고 이 두 도구를 개발한 세부 기술적인 내용이 논문으로 발표되지도 않았다.

우리도 초기에는 이미 구현된 Testground의 기술을 사용하여 P2P 네트워크 테스트 환경의 개발을 시도하였으나, 이를 위해 참고할만한 자료나 기술적 구현에 관한 내용을 쉽게 찾을 수 없었고 최신 리눅스 시스템에서 사용하기에 부적절했다. 또한 스웜 네트워크를 사용하기 위해 프로젝트를 수정하는 데에도 어려움이 있었고, 기술적 구현에 관한 내용이 연구의 성과가 우리가 아는 한 논문이나 백서의 형태로 정리되어 발표되지 않았으며, 학술적 논의가 부족하여 기존의 도구를 최신화하고 향후 연구를

위한 개선에 어려움이 있었다. 또한 개발자들의 연락처나 자문을 구하는 데에도 어려움이 있었고, 문제에 대한 응답을 얻을 수 없었다.

따라서 우리는 P2P 네트워크와 브로드캐스팅 성능 분석이 가능한 플랫폼의 제공뿐만 아니라, 최종적으로는 우리 이후에 있을 P2P 네트워크에 대한 향후 연구까지 고려하는 것을 목표로 한다. 그렇기에 우리의 K-P2PLab의 사용 방법과 아키텍처에 대한 설명, 실험의 목적과 실제 실험 환경에 따른 목적에 따라 프로그램의 수정을 도울 방법 등을 명세화하고 본 논문에서도 이에 대한 기본적인 토대를 다룰 예정이다. 그리고 여기서 더 나아가 기존의 위 연구에서 개발된 도구와 우리의 K-P2PLab을 차별화하기 위해 기존과 다른 접근 방법을 채택하였다. 우리의 도구가 기존의 도구와 가지는 차이는 한마디로 “하드웨어 효율성”으로 스웜 네트워크를 손쉽게 사용할 수 있도록 지원하여 실험자의 하드웨어 상황에 따라 최대 효율로 실험을 가능하게 하는 것이었다. 이에 대해서는 3장에서 다룰 예정이다.

III. 설계 및 구현

K-P2PLab의 최종 목표는 여러가지 브로드캐스팅 알고리즘 개선을 위한 아이디어를 실험할 수 있는 환경 개발이다. 이를 위해, K-P2PLab은 가상의 P2P 환경을 구축하고 해당 환경에서 전파 성능을 실험한 뒤, 데이터를 수집하여 P2P 네트워크를 분석하고 모니터링하는 것을 기본 목표로 하여 설계되었

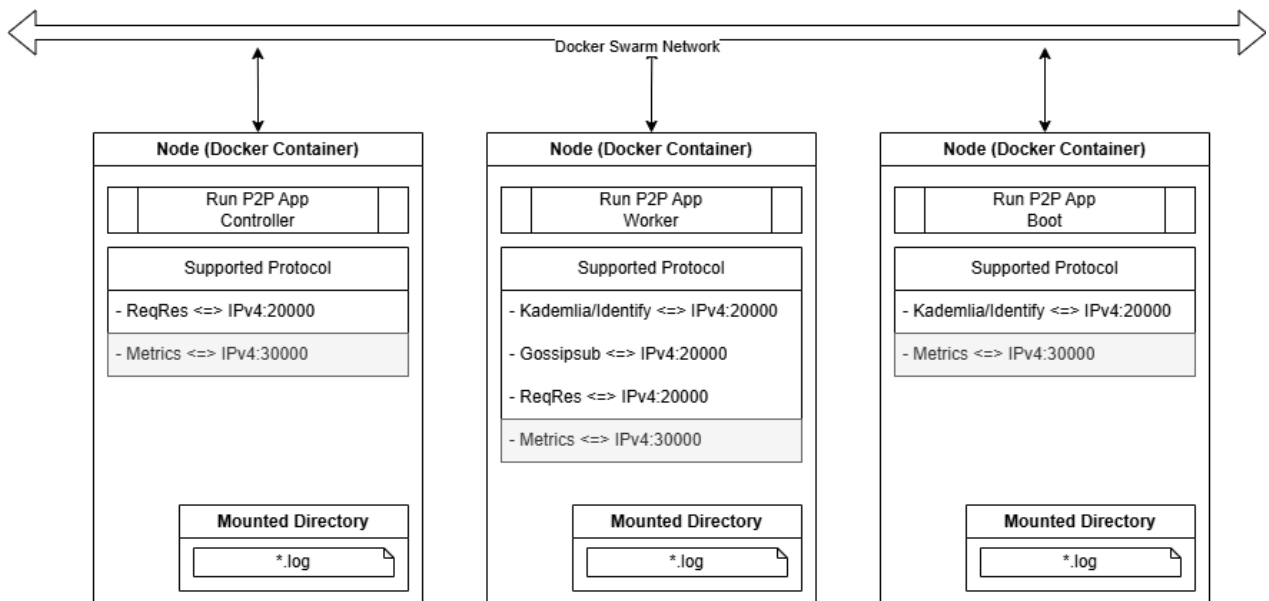


그림 2. K-P2PLab의 노드 단위 프로토콜
Fig. 2. Node-by-Node Protocol for K-P2PLab

다.

기술적으로 노드를 완전히 별도의 호스트로 분리하는 것이 가장 좋으나, 물리적 한계 및 관리의 편리함을 위해 도커의 컨테이너 기술을 사용하여 가상화 된 호스트 즉, 도커 컨테이너 상에 모든 노드를 분리하였다. 또한 P2P 피어는 현재 이더리움 네트워크 등에서 사용되고 있는 가장 대표적인 P2P 라이브러리인 libp2p를 사용하였으며[14], 이더리움의 구현과 동일하게 해당 라이브러리의 Go 언어 버전을 사용하여 P2P 어플리케이션을 구현하였다.

그림 1은 K-P2PLab의 전체적인 구조를 보여준다. 실험 환경은 전체 실험을 관리하는 하나의 Manager 서버와 확장 가능한 한 개 이상의 Runner 서버로 구축되며, 양 서버에서 Boot, Worker, Controller 3가지 형태의 피어가 동작하며 가상의 P2P 환경이 형성된다. 매니저 서버는 도커 네트워크 전체의 클러스터링 매니저 서버이며, 동시에 도커 이미지 레지스트리를 관리하고 러너 서버로부터 로그 파일을 수집한다. 그 외 Prometheus를 비롯한 필요에 따른 기타 분석 도구의 컨테이너가 동작되기도 한다. 초기 설정이 완료된 이후 매니저 서버에서 Build를 실행하게 되면, P2P 어플리케이션 도커 이미지가 빌드되어 미리 실행해둔 로컬 도커 레지스트리에 저장된다. 이 이미지는 이후 부트 노드나 워커 노드를 실행할 때 사용되며, 실행된 각 노드는 도커 스웸 네트워크를 통해 러너 서버에서 작동하게 된다. 컨트롤러 노드는 매니저 서버에서 작동하

여 워커 노드를 선택하여 퍼블리시 명령을 내릴 수 있도록 구성되어 있으며, 매니저 서버에서 실행된다. 그 외에도 워커 노드는 그룹이라는 개념과 각 그룹 안에서 노드의 번호를 부여받는데, 그룹 단위의 노드 개수 조정과 그룹별로 실험자가 원하는 행동 양식에 맞게 움직이도록 설계할 수 있다.

각 노드는 동작하며 자신의 행동에 대한 로그를 남기고 이는 매니저 서버로 수집된다. 매니저 서버에 수집된 로그는 별도의 분석 서버에서 K-P2PLab Analyzer를 통해 Metrics의 형태로 가공되며, 여기에는 시간별 P2P 네트워크의 다양한 토폴로지 정보를 알아보기 쉽게 가공하고 표현해주는 뷰어까지 포함되어 있다.

기존에 위에서 언급한 Whiteblock 및 Testground 두 도구를 비롯한 대부분의 P2P 실험 환경 생성 도구는 로그의 수집이나 노드에 브로드캐스팅 명령을 전달하는 것을 구현하기 위해 REST API를 주로 사용하였다. 이는 K-P2PLab을 구현하는 데에도 영향을 주었고, 초기에 K-P2PLab 역시 REST API를 사용한 위와 유사한 방식으로 구현되었다. 그러나, 구현의 경험상 REST API의 사용은 피어의 리소스 사용량을 늘리고 한 서버 당 노드의 최대 숫자를 감소시키는 영향이 있었다. 우리는 동일한 소형 실험 PC에서 REST API를 사용할 때와 그렇지 않을 때 같은 형태로 배포할 수 있는 최대 피어의 개수가 크게 변화하는 것을 볼 수 있었다. 실험적으로 REST API의 사용 시 최대 30개의 피어를 별도의

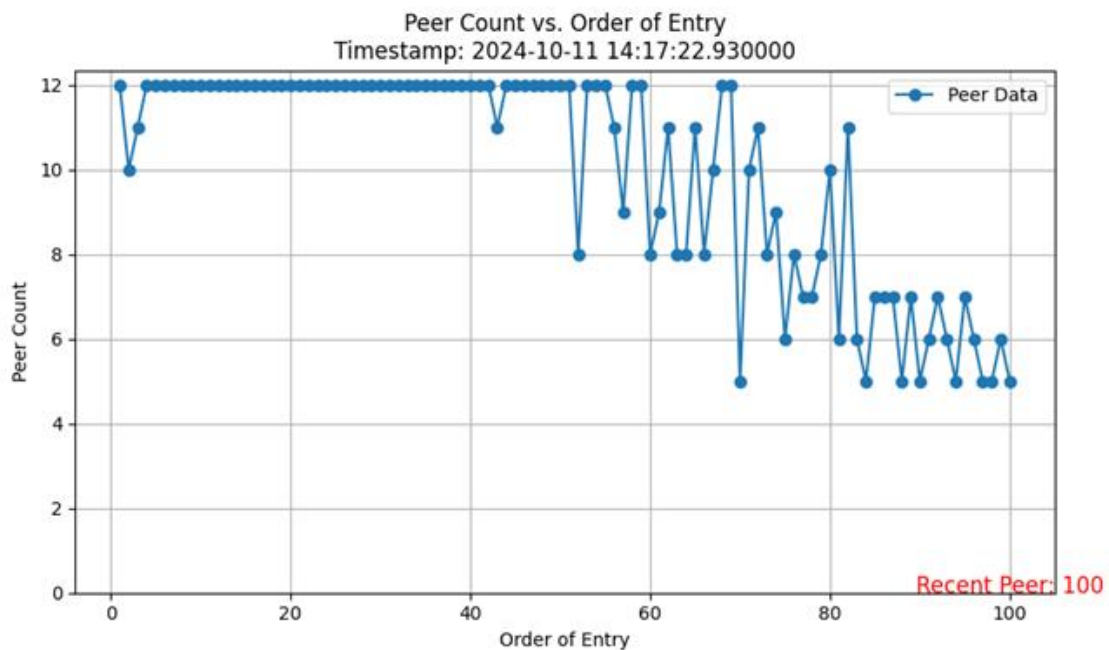


그림 3. 노드의 참여에 따른 각 노드의 차수 변화

Fig. 3. Changes in the degree of each node due to node participation

컨테이너로 가능한 위 PC에서는 REST API를 제거하자 최대 50개까지 피어를 별도의 컨테이너로 동작시킬 수 있었다. 이는 REST API를 사용하지 않는 구현이 소규모 실험자나 열악한 실험 환경에서 더 우세함을 나타내며, 이러한 환경에서의 실험에 도움이 됨을 알 수 있다. 더 많은 사람이 작은 컴퓨터로 실험이 가능해짐에 따라 더 많은 아이디어가 우리의 플랫폼에서 테스트 될 것이고, 이는 P2P 네트워크의 전파 성능을 개선하기 위한 연구를 가속하는데 기여한다.

REST API를 사용하지 않는 대신, 노드에게 전파 명령을 전달할 방법으로 libp2p 자체의 프로토콜을 사용하였다. Controller라는 libp2p 프로토콜 상에서 명령을 전달하는 별도의 피어를 두고, 해당 피어의 위치를 피어들이 사용하는 DHT[15] 네트워크와 완전히 분리시켜 각 피어가 실제 물리적 호스트에 가하는 하드웨어의 부담을 줄이면서도 네트워크에 관여하지 않는 구조를 완성하였다. 또한 각 피어가 로그를 작성함에 있어 네트워크상으로 로그를 전송하는 것이 아닌 로컬 파일로 로그를 작성하고, 이를

매니저 서버가 수집하는 방식을 채택함으로써 P2P 네트워크가 실험 외의 부가적인 환경에 의해 받는 영향을 최소화하였다.

IV. K-P2PLab 기술 사양

Go-libp2p로 개발된 각각의 실행중인 P2P 어플리케이션은 도커 컨테이너 하나당 어플리케이션 하나씩 배포되며, 각 도커 컨테이너는 도커 네트워크 위에서 고유한 IPv4 주소를 부여받고, 20000번 포트를 사용해 libp2p 통신을 진행한다. 이를 위해 Go-libp2p의 구현 기술을 적극 활용하였으며, 그 예시로 피어 검색 기술로 네트워크의 확장성이 높은 Kademlia[16]를 선택하였다. 메시지 라우팅 기술 역시 libp2p의 Gossipsub[17]이라 하는 Protocol Labs에서 개발한 기존의 Pubsub[18]에서 한층 더 발전된 기술을 적극 활용하였다. 각 피어들은 또한 30000번 포트로 자신의 Metrics 정보를 주기적으로 내보내어 이를 Prometheus와 Grafana가 수집하고 분석할 수 있다. 배포 가능한 노드의 최대 숫자는

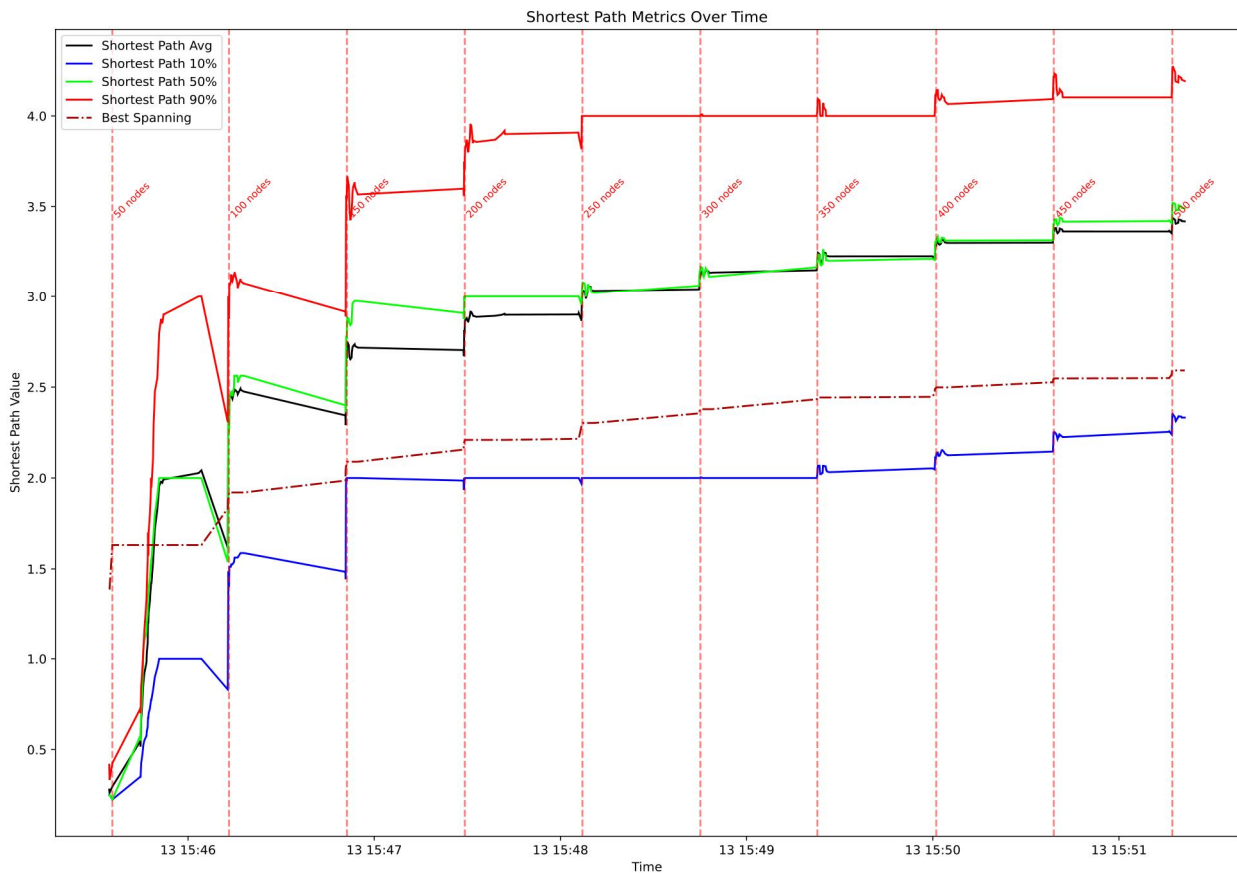


그림 4. 노드의 진입에 따른 각 노드가 갖는 Shortest Path 값 4가지의 Trimmed Average의 변화와 해당 시점에서 가장 이상적인 직경(Diameter)을 나타낸 그림

Fig. 4. Four Shortest Path value's Trimmed Average and the most ideal Diameter at that point due to participation of node

IPv4 전체로 이는 사실상 노드 배포의 한계가 존재하지 않으며, 러너 서버의 증축을 통해 간단히 실험 환경을 확장 가능하며, 서버 환경을 최대로 활용하여 실험을 진행할 수 있다.

피어는 노드가 실행될 때 역할을 부여받는데, 이는 총 3가지로 Boot, Worker, Controller가 있다. 부트 피어는 Kademia를 위한 부트스트랩 노드로, 다른 피어가 네트워크에 진입하기 위한 진입점의 역할을 한다. 부트 피어는 Gossipsub이 구현되어 있지 않으며, 따라서 메시지 전파에는 관여하지 않는다. 하지만 Metrics은 열려 있으므로 필요에 따라 정보를 수집할 수는 있다. 워커 피어는 Gossipsub이 구현된 노드로 메시지 전파에 기여하고 실제 네트워크를 구성하는 노드이다. Kademia를 사용하며 부트 피어를 통해 네트워크에 참여하고, Gossipsub을 통해 메시지를 전파한다. 컨트롤러 피어는 워커 피어 중 하나를 선택해 그 노드에게 퍼블리싱 명령을 전달할 수 있게 만들어진 피어이며, 명령을 전달 받은 워커 피어는 즉시 퍼블리싱을 진행한다. 현재는 임의의 1KiB짜리 패킷을 퍼블리싱 하도록 설계되어 있다. 또한 컨트롤러 피어는 Gossipsub은 물론 Kademia 네트워크에도 전혀 참여하지 않는다. 이는 실행 중인 컨트롤러 노드가 실험 중인 P2P 네트워크와 완전히 분리 되어있음을 의미하며, 오로지 기본적인 단일 다이얼을 통해 특정 워커 피어에게만 명령을 전달한다.

각 피어가 동작하며 발생한 동작은 로그에 기록되며, 로그는 노드 내부의 /libp2p 디렉토리에 작성된다. 이 디렉토리는 각 러너 서버마다 존재하는 /log라는 디렉토리와 마운트 되어있으며, 각 러너 서버의 /log라는 디렉토리는 다시 매니저 서버의 /log-nfs라는 디렉토리에 NFS로 공유 되어있다. 이를 통해 피어가 작성한 모든 로그 파일이 매니저 서버의 한 디렉토리에 모이는 것처럼 동작한다. 로그의 출력 수준 및 필터링은 미리 설정 파일을 통해 설정 가능하며, 그 외에도 다양한 파라미터를 미리 설정 가능하다.

또한 서버를 처음 구축하고 클러스터링 하는 방법과, 실험을 진행하는 방법이 별도의 마스터 프로그램으로 구현되어 있어 편리하면서도 세밀하게 조작이 가능하다. 앞서 말한 마스터 프로그램은 K-P2PLab을 효율적으로 사용하기 위한 CLI의 형태로 제공된다.

마지막으로 모든 피어는 네트워크상에 Metrics를 전파하도록 설계되었는데, 이를 통해 다른 외부 도구를 활용하여 네트워크를 모니터링 및 분석 가능하게 하였다.

V. K-P2PLab 검증 및 활용

우리는 앞서 설명한 우리가 개발한 도구들을 활용하여 직접 로그와 Metrics 데이터를 수집하였고, 이를 Go-libp2p의 구현 명세와 대조하여 정상적으로 작동하는 것을 확인하였다.

이 과정에서 NetworkX 기반의 분석 및 출력 도구로 우리가 앞으로 연구하고자 하는 목표를 조사해보았다. 그중 하나는 TCP 연결을 시각화한 P2P 네트워크 그래프를 출력하였을 때, 기본적인 libp2p 구현이 매우 조밀하고 불필요한 TCP 커넥션을 유지하는 것을 확인하였고, 이는 네트워크 성능에 부담을 줄 우려가 존재한다고 판단하였다. 또 다른 하나는 위 도구로 Gossipsub의 메시 네트워크를 출력함으로써 각 노드의 연결 차수가 최대 차수 D_{High} 에 근접하게 유지되는 것을 확인하였고, 피어가 최대 수치의 메시 연결을 가지면 이후의 Graft 요청을 즉시 Prune 메시지를 반환하여 거절하는 메커니즘이 포함되어 있음을 검증하였다.

그림 3은 100개까지 차례로 노드가 참여만 하고 이탈이 존재하지 않는 환경에서 노드의 진입 순서 (Order of Entry, 가로축)와 노드의 차수(Peer Count, 세로축)에 관한 그래프이다. 기존 노드들이 하나도 이탈하지 않는 극단적인 네트워크에서는 새로운 노드가 참여하여도 앞서 설명한 메커니즘으로 인해 새롭게 참여한 노드들이 기존 노드들에 비해 낮은 이상적인 차수 값 D 에 근접한 메시 연결을 가지는 것을 알아내었다. 현재 libp2p는 위 두 문제에 대해 크게 논의되지 않고 있으나, 이에 대한 자세한 연구 또한 필요할 것으로 생각된다. 100번째 피어가 들어온 시점에서 약 50번째로 들어온 피어를 기준으로 전과 후의 피어들의 전반적인 차수가 차이가 나는 것을 볼 수 있다.

그림 4는 네트워크에 노드가 참여함에 따른 P2P 네트워크의 Shortest Path 값 4가지(한 노드로부터 다른 모든 노드까지 가기 위해 거쳐야 하는 최단 노드의 수를 리스트로 만든 것을 L이라고 할 때, 정렬된 L의 각 10%, 50%, 90%째 값과 L 전체의 평균)를 노드 별로 구한 값들의 절단 평균(Trimmed

Average)을 나타낸 그래프이다. 이때 사용된 Gossipsub 파라미터 D_{Low} , D , D_{High} 는 각각 4, 6, 12인 Go-libp2p Gossipsub의 기본 설정 값이다. 또한 각 그래프 상의 점선은 네트워크의 직경이 최단 거리로 구성될 수 있는 스페닝트리 형태를 구성할 때의 직경으로, 해당 점선과 그래프 상의 다른 직선을 비교하여 현재 네트워크의 파라미터에 따른 효율을 유추할 수 있다. 절단 평균을 사용한 이유는 네트워크상에 노드가 급격히 늘어날 때 발생하는 노이즈와 같은 값을 제거하여 그래프가 시각적으로 난잡해지는 것을 막기 위함이다.

본 실험에서 절단 평균은 각 극단 값을 10%씩 제거하였고, 각 노드가 30초의 간격을 두고 50~500까지 50개씩 참여하였다. 이때 순간적으로 노드가 증가함에 따라 직경값에 심한 노이즈를 보이는 각 네트워크 확장 시간(그래프 상에서 붉은 세로 점선)을 제외하면 전체적으로 로그 스케일에 가까운 증가 추세를 보임을 알 수 있다. 또한 각 10% 값의 절단 평균을 나타내는 파란 선과 50% 값의 절단 평균을 나타내는 초록 선, 90% 값의 절단 평균을 나타내는 빨간 선이 올바르게 배치되어 있다는 사실을 통해 해당 그래프가 네트워크를 올바르게 해석하고 있음을 보이고 있다.

VI. 결론 및 향후 연구

우리는 K-P2PLab이라는 P2P 가상 실험 환경 구축 및 분석을 위한 도구를 개발하였다. 이 도구로 토폴로지 측정 및 전파 성능을 측정하여 P2P 네트워크의 성능 데이터를 충분히 수집하고 분석할 수 있음을 확인하였다. 우리는 도커 기술과 데이터 모니터링 및 네트워크 분석 기술을 활용하였다.

현재 K-P2PLab은 노드 간 연결에 대한 통신 성능을 조절하는 기능이 구현되어 있지 않다. 실제 P2P 네트워크와 동일한 환경에서 실험을 진행하기 위해서는 통신성능을 조절하는 기능이 필요하며 Linux의 TC(Traffic Control) 기능을 활용하여 노드 간 통신성능(전파 시간, 패킷 손실 등)을 조절할 수 있게 발전시킬 계획이다. 이후에는 K-P2PLab을 활용하여 여러 브로드캐스팅에 관련된 성능 실험하고 성능을 높일 수 있는 다양한 방법을 고안할 것이며, 다른 연구자들이 브로드캐스팅 성능 개선을 위한 아이디어를 실험하고 분석하기 위한 도구로 우리의 플랫폼이 사용될 수 있을 것이다. 또한 위에서 언급한 TCP 커넥션 및 Gossipsub 메시 연결 문제에 대

해, TCP 커넥션을 평가하고 조절하여 최대 개수를 제한하는 알고리즘 혹은 프로토콜의 개발에 관한 연구와 주기적 메시 연결 정리를 통해 정적인 P2P 네트워크상의 균등한 차수를 갖게 하는 프로토콜 도입 등의 방법을 통해 해결하는 것도 주요한 향후 연구 주제로 다루어질 수 있다.

References

- [1] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," 2008.
- [2] Buterin, V., "Ethereum white paper," GitHub repository, vol. 1, pp. 22-23, 2013.
- [3] Werth, J., Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N., and Pahl, C., "A review of blockchain platforms based on the scalability, security and decentralization trilemma," in Proc. ICEIS (1), 2023, pp. 146-155.
- [4] Saad, M., Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D., and Mohaisen, D., "Exploring the attack surface of blockchain: A comprehensive survey," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1977-2008, 2020.
- [5] 김정연, 주홍택, "코스모스에서 블록체인 간 메시지 전달 시간 측정 및 분석," 한국통신학회논문지, vol. 47, no. 10, pp. 1556-1564, 2022, doi: 10.7840/kics.2022.47.10.1556.
- [6] 김애리, 주홍택, "비트코인 네트워크에서 압축 블록 전달 방식의 전송 지연 분석," 한국통신학회 논문지, vol. 47, no. 6, pp. 826-835, 2022, doi: 10.7840/kics.2022.47.6.826.
- [7] 맹수훈, 주홍택, "이더리움 네트워크에서 액티브 프루빙을 이용한 정확도 높은 토폴로지 탐색 방법," 한국통신학회논문지, vol. 47, no. 3, pp. 453-460, 2022, doi: 10.7840/kics.2022.47.3.453.
- [8] libp2p, "libp2p/go-libp2p," GitHub. Available: <https://github.com/libp2p/go-libp2p>. [Accessed: Nov. 20, 2024].
- [9] Docker, "Docker: Accelerate how you build, share, and run modern applications," Available: <https://www.docker.com/>. [Accessed: Nov. 20, 2024].

- [10] Prometheus, "Prometheus: Monitoring system & time series database," Available: <https://prometheus.io/>. [Accessed: Nov. 20, 2024].
- [11] Grafana Labs, "Grafana: The open observability platform," Available: <https://grafana.com/>. [Accessed: Nov. 20, 2024].
- [12] Whiteblock, "Whiteblock GitHub repositories," GitHub. Available: <https://github.com/whiteblock>. [Accessed: Nov. 20, 2024].
- [13] Testground, "Testground GitHub repositories," GitHub. Available: <https://github.com/testground>. [Accessed: Nov. 20, 2024].
- [14] libp2p, "libp2p and Ethereum," Available: <https://blog.libp2p.io/libp2p-and-ethereum/>. [Accessed: Nov. 20, 2024].
- [15] Dabek, F. F. E., "A distributed hash table," Doctoral dissertation, Massachusetts Institute of Technology, 2005.
- [16] Maymounkov, P., and Mazieres, D., "Kademlia: A peer-to-peer information system based on the xor metric," in Proc. Int. Workshop Peer-to-Peer Syst., Springer Berlin Heidelberg, 2002, pp. 53-65.
- [17] Vyzovitis, D., and Psaras, Y., "Gossipsub: A secure pubsub protocol for unstructured, decentralised p2p overlays," Proc. Protocol Labs TechRep, PL-TechRep-gossipsub-v0., 2019.
- [18] Banerjee, T., and Sahni, S., "Pubsub: An efficient publish/subscribe system," IEEE Trans. Comput., vol. 64, no. 4, pp. 1119-1132, 2014.

이 성 옥 (Sung-wook Lee)



2021년 3월~현재: 계명대학교
컴퓨터공학과 학사 과정

<관심분야> 컴퓨터 공학, 네트워크, 블록체인
[ORCID: 0009-0002-5796-6474]

주 흥 택 (Hong-taek Ju)



1989년 2월: KAIST 컴퓨터공
학과 졸업
1991년 2월: POSTECH 컴퓨
터공학과 석사
2002년 2월: POSTECH 컴퓨
터공학과 박사
2002년 9월~현재: 계명대학교

컴퓨터공학과 교수
<관심분야> 컴퓨터 공학, 네트워크, 블록체인
[ORCID: 0000-0002-8434-485X]