JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 23, No. 9, pp. 1699-1711, Sep. 2022

Check for updates

AMB 기반의 신뢰적 배송추적 서비스 플랫폼 개발

최 환 석[1] · 오 지 훈[2] · 이 주 형[3] · 윤 영 주[2] · 이 우 섭[4*]
[1]카이스트-메가존클라우드 지능형 클라우드 융합기술 연구센터 매니저 [2]국립한밭대학교 정보통신공학과 학사
[3]아주대학교 정보통신대학원 석사과정 [4*]국립한밭대학교 지능미디어공학과 교수

Development of AMB based Reliable Delivery Tracking Service Platform

Hoan-Suk Choi[1] · Ji-Hoon Oh[2] · Ju-Hyeong Lee[3] · Young-Joo Yoon[2] · Woo-Seop Rhee[4*]

[1]Manager, KAIST-Megazone Cloud Intelligent Cloud Computing Convergence Research Center, Daejeon 34141, Korea
[2]Bachelor's degree, Department of Information Communication Engineering, Hanbat National University, Daejeon 34158, Korea
[3]Marster's degree course, Graduate School of Information Communication Technology, Ajou University, Suwon 16499, Korea
[4*]Professor, Department of Intelligent Media Engineering, Hanbat National University, Daejeon 34158, Korea

[요 약]

코로나19로 인해 급증한 온라인 쇼핑 및 배송으로 파손, 분실 등 관련 피해가 함께 증가하고 있다. 하지만 과실책임 입증에 관한 규정 및 방법이 부족하여 추가적인 분쟁 피해가 발생하고 있다. 따라서 본 논문은 IoT, 클라우드 컴퓨팅, 블록체인 기술을 활용한 배송추적 서비스 플랫폼을 제안한다. IoT 센서로 수집되는 데이터는 클라우드에 저장되며, 블록체인 기반의 스마트 컨트렉트를 통해 특정 이벤트 정보를 블록체인 네트워크에 기록한다. 또한 DApp을 제공하여, 생산자, 소비자, 배송자가 배송과정을 객관적으로 모니터링 관리할 수 있도록 하여, 배송서비스의 신뢰도를 향상시킨다. 제안하는 서비스 플랫폼은 대표적 클라우드 컴퓨팅 서비스인 AWS의 EC2, AMB 등을 활용하여, 안전하고 효율적인 클라우드 인프라를 기반으로 구축되었다.

[Abstract]

Due to COVID-19, online shopping and delivery have soared, and related damages are also increasing. However, there is a lack of regulations and methods for proving negligence, resulting in additional dispute damage. Therefore, this paper proposes a delivery tracking service platform using IoT, cloud computing, and blockchain technology. Data collected by IoT sensors is stored in the cloud, and specific event information is recorded on the blockchain network through blockchain-based smart contact. In addition, DApp is provided to producers, consumers, and deliverers for improving the reliability of delivery services by objectively monitoring and managing the delivery process. The proposed service platform was built based on a safe and efficient cloud infrastructure using AWS' EC2 and AMB, which are representative cloud computing services.

색인어 : 블록체인, 신뢰, 배송추적 플랫폼, 아마존 관리형 블록체인, 클라우드 컴퓨팅

Keyword : Blockchain, Trust, Delivery Tracking Platform, Amazon Managed Blockchain, Cloud Computing

http://dx.doi.org/10.9728/dcs.2022.23.9.1699

Received 09 August 2022; **Revised** 06 September 2022
Accepted 06 September 2022

***Corresponding Author; Woo-Seop Rhee**

Tel: +82-42-821-1749
E-mail: wsrhee@hanbat.ac.kr

Ⅰ. 서 론

코로나 19로 인해 언택트 시대가 가속화되면서 일반 상품을 온라인으로 구입하는 것뿐만 아니라 식품 및 생필품의 온라인 구매가 급증하였고 새벽배송 및 당일배송 등의 다양한 배송서비스가 도입되었다. 대표적 온라인 마켓인 지마켓과 옥션은 온라인 장보기가 2018년 1~9월 대비 2021년 동기간에 43% 증가했으며[1], 2020년 택배 물량은 전년도 대비 20.9%가 폭증한 33억 7천만 개에 이르며 최대 증가 폭을 보이고 있다. 하지만 유통 물량이 증가하면서 파손, 분실 등 관련 피해가 함께 증가하고 있다. 소비자가 피해보상을 신청하면 공정거래위원회 택배 표준 약관에 따라 택배사가 우선 배상하게 되어 있지만, 과실책임 입증에 관한 규정 및 방법이 없어, 택배기사, 대리점, 택배회사, 판매자, 구매자 간 분쟁이 발생한다[2]. 따라서 신뢰적인 방법으로 배송과정을 추적하는 방법이 필요하다.

본 논문은 IoT 기술을 활용하여 배송 과정의 데이터를 추적하고 이를 블록체인에 저장하여 객관적 신뢰성을 가지는 서비스를 제안한다. 블록체인 기술은 분산된 네트워크 구성원이 체인 형태로 무수히 연결된 블록에 관리 대상 데이터를 저장하여, 누구도 임의로 수정할 수 없고 누구나 해당 데이터를 열람할 수 있으므로, 신뢰성이 요구되는 시스템에 활용되고 있다[3]. 하지만 블록체인을 도입하기 위해서는 고비용의 인프라를 구축해야 하며, 키 관리 및 각종 보안 이슈 등 특유의 복잡한 개발 과정이 요구된다. 따라서 제안하는 서비스는 Amazon Web Service (AWS)의 Blockchain As A service (BaaS)인 Amazon Managed Blockchain (AMB)을 기반으로 구축되었다.

AWS는 아마존닷컴의 클라우드 컴퓨팅 사업부로, 컴퓨팅, 컨테이너, 스토리지, 데이터베이스, 블록체인, Machine Learning, 분석, 보안, 비용 관리, 사물 인터넷 등 다양한 분야의 서비스를 제공하고 있으며, 이중 AMB는 클라우드 환경에 블록체인 프레임워크를 설치하고, 클릭 몇 번으로 블록체인 네트워크를 생성, 관리할 수 있도록 지원하는 관리형 서비스이다[4]. 이를 활용함으로써, 핵심 솔루션 개발에 집중할 수 있다.

따라서 본 논문에서는 AWS의 BaaS인 AMB 기반의 신뢰적 배송추적 서비스 플랫폼을 제안한다. 이를 위해 본 논문은 2장에서 AWS 기반 블록체인 서비스의 특징을 소개하고, 3장에서는 AMB 기반의 신뢰적 배송추적 서비스 플랫폼을 제안한다. 제안하는 플랫폼은 IoT 센서 노드를 통해 배송과정의 데이터를 수집하며, 스마트 컨트렉트를 통해 자동으로 블록체인 네트워크에 중요 정보를 업로드 한다. 또한 생산자, 소비자와 배송자는 DApp을 이용하여 상품을 등록하고, 배송과정을 갱신하며, 실시간 정보 및 이상상황 발생 여부를 확인할 수 있다. 4장에서는 제안하는 서비스 플랫폼의 구현 결과를 제시하고 5장에서 결론을 맺는다.

Ⅱ. AWS 기반 블록체인 서비스

블록체인 기술을 활용할 경우 신뢰할 수 있는 중앙 기관이 없어도 여러 참여자가 트랜잭션을 실행할 수 있는 신뢰적 애플리케이션을 구축할 수 있다. 현재 기존 기술을 사용하여 확장 가능한 블록체인 네트워크를 구축하려고 하면 설정과 관리가 매우 복잡하다. 블록체인 네트워크를 생성하려면 각 네트워크 멤버가 수동으로 하드웨어를 프로비저닝하고, 소프트웨어를 설치하며, 액세스 제어를 위한 인증서를 생성 및 관리하고, 네트워킹 구성 요소를 구성해야 한다. 블록체인 네트워크를 실행한 후에도 지속적으로 인프라를 모니터링하고 변경되는 상황 (예: 트랜잭션 요청 증가, 네트워크의 신규 멤버 가입 및 탈퇴)에 맞게 관리 및 조정해야 한다[5].

그러나, AWS에서 제공하는 완전 관리형 블록체인 서비스인 AMB를 사용할 경우, 블록체인 인프라를 셀프 호스팅하는 방식과 달리, AMB는 자동으로 하드웨어를 프로비저닝하고, 소프트웨어를 구성하며, 네트워킹 및 보안 구성 요소를 설정할 필요가 없다. 네트워크 참가자는 AMB의 투표 API를 사용하여 투표를 진행해 멤버를 추가하거나 제거할 수 있다. 추가된 멤버는 여러 블록체인 피어 노드를 시작 및 구성하여 트랜잭션 요청을 처리하고 원장 복사본을 저장할 수 있다. 또한 AMB는 네트워크를 모니터링하여 성능이 낮은 노드를 자동으로 대체한다[6].

AMB는 블록체인 오픈 소스 프레임워크인 하이퍼레저 패브릭 (Hyperledger Fabric)과 이더리움 (Ethereum)을 지원하여 클릭 몇 번으로 퍼블릭 네트워크에 참여하거나 확장 가능한 프라이빗 네트워크를 설정 및 관리할 수 있는 완전관리형 서비스로써, 네트워크를 생성하거나 퍼블릭 네트워크에 참여하는 데 필요한 오버헤드를 제거하며, 수백만 건의 트랜잭션을 실행하는 애플리케이션 수천 개의 수요에 맞춰 자동으로 확장된다[7]. 트랜잭션 생성 및 검증을 위한 추가 용량이 필요한 네트워크 멤버는 API를 사용하여 새 피어 노드를 빠르게 추가할 수 있으며 AWS는 CPU 및 메모리 조합으로 구성된 다양한 인스턴스 유형을 지원하여, 워크로드에 적합한 리소스 조합을 선택할 수 있다. 또한, AWS Key Management Service (KMS) 기술을 사용하여 네트워크 인증서를 보호 및 관리하여 자체적인 보안 키 스토리지가 요구되지 않고 새 멤버를 손쉽게 초대할 수 있다[8].

또한 블록체인 네트워크의 트랜젝션 추적 및 관리를 제공하는 오더링 서비스(Ordering Service)의 안정성 개선을 위해 Amazon QLDB를 제공한다. 하이퍼레저 패브릭의 오더링 서비스는 전체 트랜잭션 기록을 저장하지 않아 트랜잭션 기록의 추적 및 복구가 어렵다. Amazon QLDB는 완전 관리형 원장 데이터베이스로, 중앙의 신뢰 기관이 소유하는 투명하고, 변경 불가능하며, 암호화 방식으로 검증 가능한 트랜잭션 로그를 제공한다. 이를 기반으로 모든 애플리케이션 데이터 변경 내용을 추적하며 완전하고 검증 가능한 시간별 변경 기록을 유지할 수 있으며, 변경 기록 요약(예: 상품의 모든 이전

소유자 목록) 및 트랜잭션 기록과 관련된 세부 정보(예: 상품 판매 시간 및 새 소유자의 이름)도 질의 할 수 있다[9]. 이는 자동으로 스토리지 및 리소스를 조정하는 서버리스 구조로 되어 있어 소규모로 적용할 수 있으며, 사용한 만큼만 요금을 지불하는 장점이 있다.

Ⅲ. 제안하는 AMB 기반 신뢰적 배송추적 서비스 플랫폼

배송 서비스란 생산과 소비기능 사이에서 생산물 또는 생산품 등을 적절히 이동시키는 과정을 제공하는 서비스로 가장 대표적인 예로 택배를 들 수 있다. 그러나 지금의 배송 서비스는 생산물/생산품이 배송되는 과정 중 파손 및 분실의 발생된 원인을 알기 어렵고, 이에 대한 배상 및 책임 소재를 판정의 문제를 가지고 있어 최종 배달 과정의 택배 기사에게 그 책임이 전가 되는 등 사회적 문제로 대두되고 있다. 또한 생산/가공지를 허위로 작성하여 소비자를 속이는 사례도 발생하면서 가공/생산, 배송, 소비자간 서비스의 신뢰에 좋지 않은 영향을 미치고 있다.

이를 위해 최근 위조, 변조가 어렵고 분산 원장으로 데이터를 저장하여 탈 중앙화 및 데이터 분산의 장점을 지닌 블록체인의 기술을 배송 서비스에 활용하여 생산/가공부터 소비자가 받는 시점까지의 모든 배송 서비스 데이터를 저장하고 신뢰성 문제점을 해결하는 연구들이 진행되고 있다[10].

그림 1은 본 논문에서 제안하는 AMB 기반 신뢰적 배송 추적 서비스 구성도이다. 배송중 발생되는 데이터 수집을 위해 IoT 센서를 상품에 설치하며, 수집된 데이터는 AWS 클라우드 기반 DB인 RDS[11]에 저장된다. 이중 지속적으로 불변 형태로 유지해야 할 중요 정보 및 특정 이벤트는 스마트 컨트렉트 (Smart contract)를 통해 조건 만족시 자동적으로 Amazon AMB에 업로드 한다.

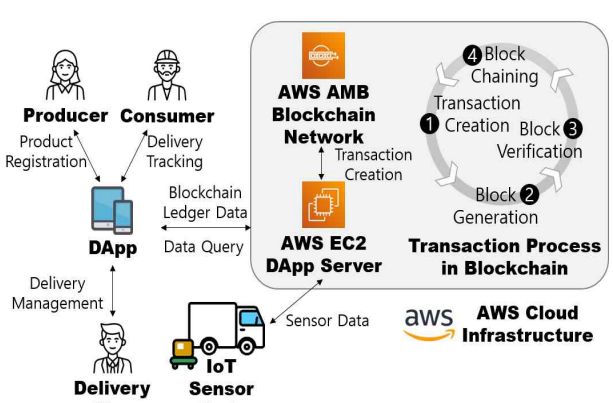

그림 1. 제안하는 AMB 기반 신뢰적 배송추적 서비스 구성도
Fig. 1. Overview of the proposed AMB based reliable delivery tracking service

이와 같이 RDS 및 AMB에 저장된 다양한 배송 서비스 정보는 생산자, 소비자, 배송업체에게 DApp을 통해 활용된다. 서비스 사용자는 자신의 상품을 등록하고, 배송 과정의 정보를 입력하며, 자신이 주문한 상품의 현재 상황을 파악 하는 등 배송과정의 전반적 정보를 관리하고 제공받는다. 만약 배송과정에서 파손, 변질과 같은 심각한 이벤트가 발생하였다면, 이는 변조 불가능한 AMB에 저장되어, 추후 책임소재 판정의 정보로 활용될 수 있다.

3-1 신뢰적 배송 추적 서비스 플랫폼 구조

본 논문에서는 다음 그림2와 같이 AWS에서 제공하는 대표적인 인스턴스인 EC2, RDS를 활용하고, AMB의 하이퍼레저 블록체인을 이용하여 신뢰성 있는 배송추적 서비스를 제공할 수 있는 플랫폼 구조를 제안한다.

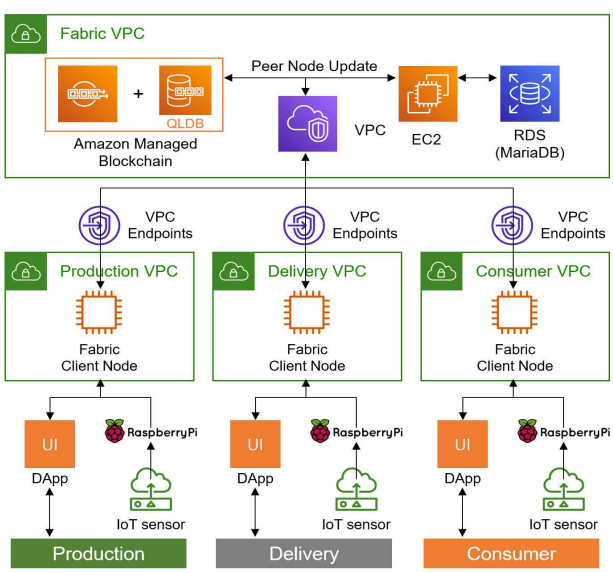

그림 2. 제안하는 AMB 기반 배송추적 서비스 구조
Fig. 2. Architecture of the proposed AMB based delivery tracking service

먼저 시스템의 보안을 위해 용도별로 VPC를 설정하였다. AMB 및 코어 인스턴스가 배치되어 있는 영역을 패브릭(Fabric) VPC로 정의하였고, 이는 VPC 게이트웨이를 통해 타 VPC와 통신할 수 있도록 한다. 또한 생산/가공, 배송 및 소비자와 같이 각 응용 영역별로 별도의 VPC를 설정하여 각자의 보안 설정을 유지하도록 하였으며, VPC 엔드포인트(Endpoint)를 통해 코어 영역인 패브릭 VPC의 VPC 게이트웨이와 연결된다[12].

각각의 응용단에는 라즈베리파이 기반의 IoT 센서가 설치되어 각종 데이터를 수집하고 이를 클라우드에 업로드 한다. 생산/가공자는 DApp과 IoT센서를 통해 생산/가공 과정에서 발생하는 정보나 물품의 현재 상태 데이터를 수집 및 입력하

여 트랜잭션의 형태로 RDS서버의 MariaDB에 저장한다.

배송 분야에서는 운반중인 차량에 탑재된 IoT센서를 통해 상품의 현재 상태나 위치정보 데이터를 RDS서버의 MariaDB로 전달하며, DApp을 통해 실시간으로 상품의 상태를 확인하고 문제에 대해 즉각 대응할 수 있다.

가공/생산자와 배송중 RDS서버로 전달된 트랜잭션은 EC2환경에서 체인코드(Smart contract)를 통하여 AMB네트워크의 피어 노드에 업데이트 한다. 이후 VPC 엔드포인트를 이용하여 각 패브릭 클라이언트 노드와 통신한다.

최종적으로 소비자는 DApp을 통하여 실시간으로 배송 서비스 전 과정에서 상품의 상태와 정보를 확인할 수 있어 가공/생산, 배송, 소비자 등 배송 서비스의 구성원간 신뢰를 증가시킬 수 있다.

3-2 제안된 구조에서 신뢰적 배송추적 서비스 절차

이 장에서는 3-1장에서 제안된 구조에서 신뢰적 배송 서비스를 위해 AMB의 하이퍼레저 패브릭을 기반으로 하는 회원가입 및 로그인, 상품 등록, 상품선택, 배송 출발 및 완료, 소비자 조회, 배상문의, 실시간 조회 절차를 제안한다.

1) 분산원장 기반 회원가입 및 로그인 프로시저

그림 3은 제안하는 회원가입 및 로그인 절차이다. 소비자는 상품조회만 가능하면 되기 때문에 꼭 회원가입이 필요하지 않지만, 생산자는 상품을 등록하고, 배송업체는 배달할 상품을 선택하여 배송출발 및 완료와 같이 상태정보를 수정해야 하며, 담당자 정보가 필요하므로 회원가입 및 로그인 절차가 필요하다. 생산자와 배송업체는 DApp을 통해 회원가입을 하면 회원 정보를 키 값 형태로 변환하고 AMB의 하이퍼레저 패브릭에 데이터유형에 맞게 블록 형태로 저장을 한다. 로그인시, 사용자는 아이디와 비밀번호를 입력하여 EC2서버에 전달하면, 대응되는 키 값의 요청주소로 데이터를 요청하여 입력한 데이터와 블록의 값을 비교하여 일치하였을 시, 유효한 접근으로 보고 로그인을 허용한다.

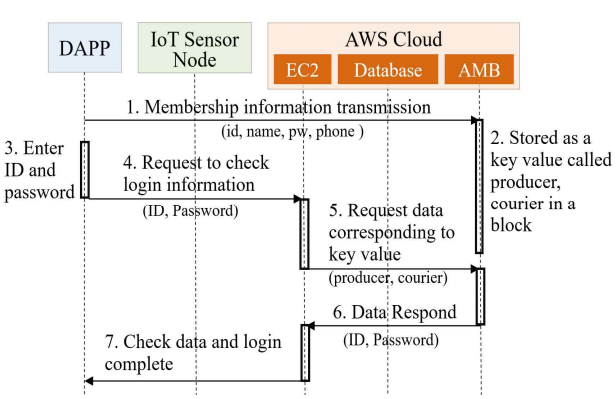

그림 3. 분산원장 기반 회원가입 및 로그인 프로시저
Fig. 3. Distributed ledger based join and login procedure

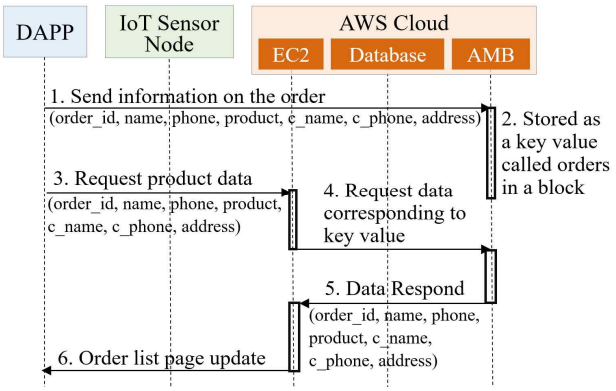

그림 4. 분산원장 기반 상품등록 프로시저
Fig. 4. Distributed ledger based registration procedure

2) 분산원장 기반 상품 등록 프로시저

그림 4는 제안하는 상품 등록 절차이다. 생산자가 DApp을 통해 로그인 후 상품을 등록하면, 상품 데이터가 회원가입 및 로그인 절차와 동일하게 키 값으로 변환되어 하이퍼레저 패브릭에 블록 형태로 저장 한다. 로그인 절차와 마찬가지로 DApp을 통해 상품 정보를 요청했을 경우 키 값에 해당되는 요청 주소로 하이퍼레저 패브릭에 블록 값을 검색하게 되며, 해당 값을 회신받은 DApp은 화면을 갱신하여 상품 정보를 보여준다.

3) 분산원장 기반 상품 선택, 배송 출발 및 완료 프로시저

그림 5는 제안하는 상품선택, 배송출발 및 완료 절차를 보여준다. 배송업체가 DApp에 로그인 하면 EC2서버가 배송되어야 할 상품 목록을 제공한다. 상품 목록에서 본인이 배달할 상품들을 선택하고, 배송 출발 버튼을 누르면 바로 다음 페이지에서 선택한 상품목록 정보를 제공하고, Maria DB에 저장된 해당 상품의 상태정보 값을 false에서 true로 변경한다. 이 정보가 변경되면 해당 상품에 부착된 IoT 센서노드에 연결된 온습도 센서가 동작을 시작하며 시간, 온도, 습도 및 이상여부 데이터를 Maria DB에 지속적으로 업로드 한다. 배송업체가 배달을 완료했을 경우, DApp의 배송 완료 버튼을 누르면 Maria DB의 상품 상태정보 값이 다시 false로 바뀌고 IoT 센서노드의 온습도 센서가 동작을 멈춘다.

4) 분산원장 기반 소비자 조회, 배상 문의 프로시저

그림 6은 제안하는 소비자 조회, 배상문의 절차이다. 소비자가 주문번호로 EC2 서버에 상품 조회 요청을 하면, 주문번호와 일치하는 상품의 상태 블록 값을 하이퍼레저 패브릭에서 가져와 제공한다. 배상문의 프로시저는 상품에 문제가 생겼을 시, 배상 문의 버튼을 누르면 EC2 서버가 해당 상품의 상태정보 중 이상여부를 나타내는 필드가 true인 데이터를 하이퍼레저 패브릭에 요청하여 그와 관련된 상품 데이터 및 담당자의 데이터를 DApp에 제공한다. 사용자는 자신의 상품에 이상상황이 발생한 데이터가 존재한다면, 제공받은 담당자 연락처를 통해 배상을 요청할 수 있다.

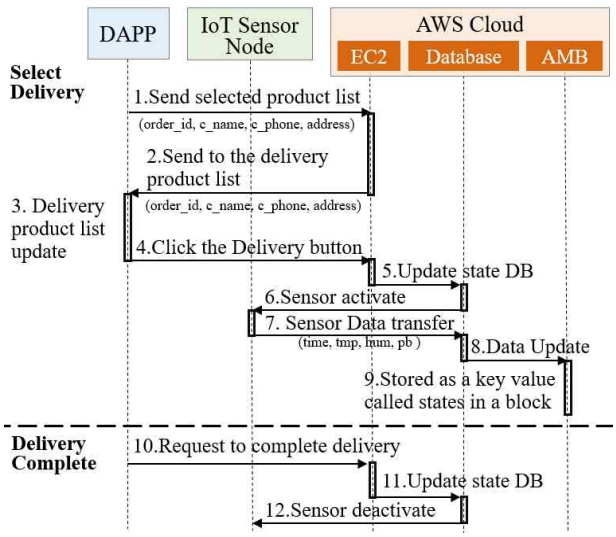

그림 5. 분산원장 기반 배송상품선택 및 배송완료 프로시저
Fig. 5. Distributed ledger based delivery select and complete procedure

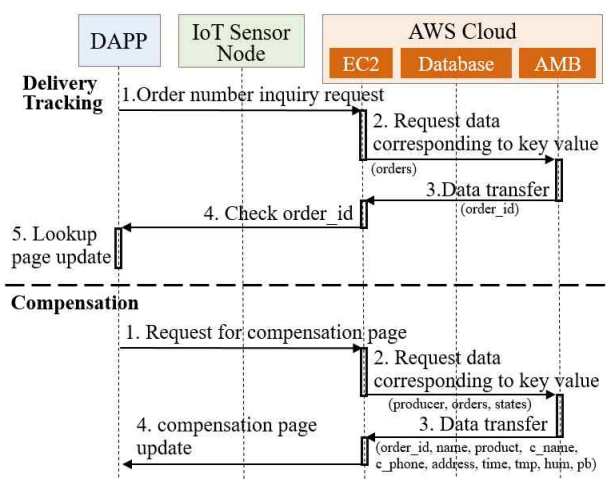

그림 6. 분산원장 기반 소비자 조회, 배상문의 프로시저
Fig. 6. Distributed ledger based delivery tracking and compensation procedure

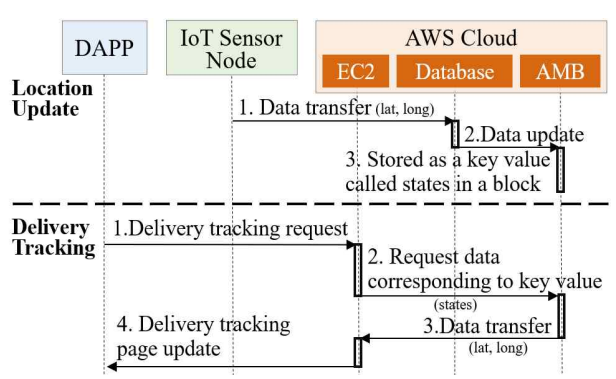

그림 7. 분산원장 기반 실시간 조회 프로시저
Fig. 7. Distributed ledger based delivery tracking and location update procedure

5) 분산원장 기반 실시간 조회 프로시저

그림 7은 제안하는 실시간 조회 절차 이다. 소비자가 실시간 조회를 요청했을 시, 온습도 센서와 마찬가지로 GPS 센서가 동작을 시작하며 위도, 경도 데이터를 Maria DB에 업로드한다. EC2 서버는 Maria DB에 저장된 위치 데이터들을 하이퍼레저 패브릭에 블록형태로 저장을 하고, 실시간 조회 요청이 들어왔을 때 EC2 서버가 이 정보를 하이퍼레저 패브릭에서 조회하여 DApp을 통해 사용자에게 알려준다.

Ⅳ. 제안하는 AMB 기반 신뢰적 배송추적 서비스 플랫폼 구현

제안하는 서비스는 AMB를 기반으로 블록체인 네트워크를 구성하여, 배송과정에서 무결성이 보장되어야 할 중요 정보를 보관한다. 이를 위해 AMB의 하이퍼레저 패브릭를 사용하며 체인코드(Smart Contract)를 통해 블록에 정보를 저장하고 이는 AMB를 통해서 모니터링 가능하다. 또한, AWS EC 2 기반 DApp 서버를 구축하였으며 Linux 환경에서 Node.js, Vue.js, Express.js 기반으로 사용자 인터페이스를 제공한다. 생산자와 소비자는 DApp을 통해 배송 상품을 등록하고 등록된 상품이 배송되는 과정을 조회할 수 있고 배송업체는 DApp에 등록된 상품을 선택해 배송을 시작한다. 라즈베리파이 기반의 IoT 센서 노드를 상품 또는 배송과정에 부착하여 배송과정의 다양한 데이터를 수집하며, AWS RDS를 기반으로 구축된 데이터베이스에 정보를 저장한다.

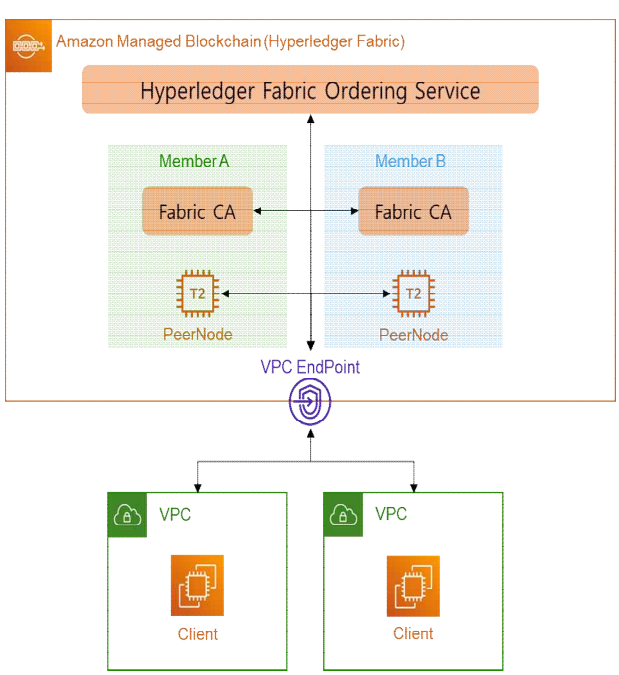

그림 8. AMB 네트워크 구성도
Fig. 8. AMB network configuration diagram

그림 8은 AMB 네트워크 리소스 구성도를 보인다. A와 B 두 개의 멤버를 T2타입의 EC2 인스턴스로 구성하였으며, 블록체인 네트워크에 접근하기 위한 VPC 엔드포인트를 설정하였다. 서비스 정보를 수집하는 IoT 노드는 패브릭 클라이언트에 연결되어 해당 정보를 블록체인 네트워크에 전달할 수 있도록 구성되었다.

4-1 AMB 기반 하이퍼레저 패브릭 네트워크 구성

AMB기반의 하이퍼레저 패브릭을 구성하기 위해 먼저 블록체인 프레임워크를 선택하여 블록체인 네트워크를 생성하고, 네트워크에 참여할 멤버와 멤버가 갖게 될 피어노드를 구성한다[13].

1) AMB 기반 블록체인 네트워크 생성

AMB에서 지원하는 하이퍼레저 패브릭 프레임워크는 프라이빗 체인 네트워크 이므로 이를 선택하기 위해 먼저 블록체인 네트워크 배너의 '프라이빗 네트워크 생성' 을 선택한다. 프라이빗 네트워크 생성을 위해서는 블록체인 프레임워크 버전, 네트워크 에디션, 네트워크 이름 및 설명, 투표 정책, 멤버 구성, 하이퍼레저 패브릭 Certificate Authority(CA), 로깅 등을 설정한 후 '네트워크 및 멤버생성' 을 선택한다.

그림 9는 위와 같은 방법으로 생성된 네트워크의 세부 정보 화면이다. 생성된 네트워크의 ID, Amazon Resource Number(ARN), 합의를 위한 투표 정책 VPC 엔드포인트, 서비스 엔드포인트 및 프레임워크 버전 등의 다양한 세부 정보를 제공하고 있다[14].

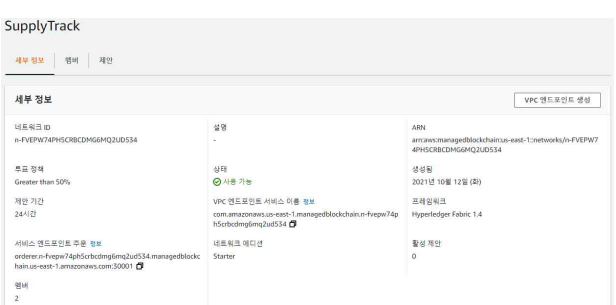

그림 9. AMB 기반으로 생성된 프라이빗 네트워크 세부 정보
Fig. 9. Detail information of AMB based private network

2) 블록체인 네트워크 멤버 추가

프라이빗 네트워크는 허용된 대상만을 네트워크를 구성하는 멤버로 인정하므로 여러 노드를 포함하는 네트워크를 구성하기 위해서는 블록체인 네트워크의 멤버를 추가해야 한다.

기존에 생성된 네트워크를 선택하면 네트워크 세부정보, 멤버정보, 제안구성 등의 기능이 제공된다. AMB 네트워크 세부 설정 화면에서 '멤버' 버튼을 선택하여 다른 멤버를 추가하기 위한 초대 제안을 생성한다.

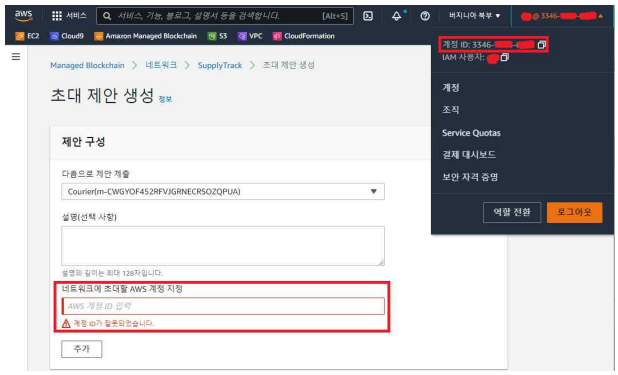

그림 10. AMB 기반 블록체인 네트워크 멤버 추가
Fig. 10. Add member for AMB based Blockchain network

그림 10은 초대 제안을 생성하는 화면으로 AWS 계정 ID 입력 필드에 초대할 상대의 12자리 계정 ID를 입력하여 현재 생성한 네트워크로 상대방을 초대할 수 있다.

초대를 받은 계정에서 AMB 서비스 화면 우측의 초대 메뉴를 선택하면, 그림 11과 같이 현재 초대받은 네트워크의 정보를 확인할 수 있다. 여기서 초대 수락 버튼을 통해 아주 간단하게 해당 프라이빗 네트워크에 멤버로 참여할 수 있다.

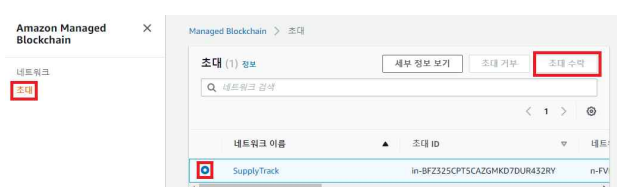

그림 11. AMB 기반 블록체인 네트워크 멤버 초대 수락
Fig. 11. Member invitation accept for AMB based Blockchain network

멤버가 추가되면 네트워크를 구성할 피어노드를 생성해야 한다. 그림 12는 AMB 네트워크의 멤버 구성을 확인하는 화면으로 피어를 생성할 특정 멤버를 선택하여 간단히 피어노드를 생성할 수 있다.

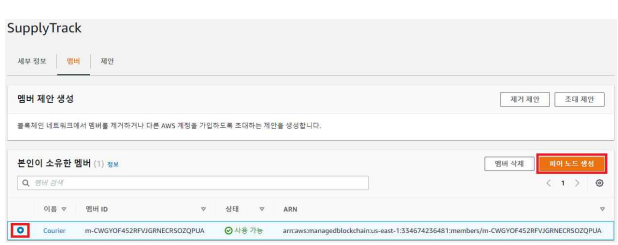

그림 12. AMB 네트워크 멤버 구성 화면
Fig. 12. AMB Network member configuration details

그림 13은 피어노드 생성을 위한 단계를 보여준다. 먼저 해당 노드가 작동할 인스턴스 유형을 선택한다. 본 논문에서는 bc.t3.small 타입의 인스턴스를 선택하였다. AWS는 다양한 프로세서, 스토리지, 메모리 등의 조합으로 이루어 진 500개가 넘는 인스턴스를 제공한다. 이를 통해 범용 타입 뿐만

아니라 컴퓨팅, 메모리, 가속화, 스토리지 등에 최적화 된 인스턴스를 용도에 맞게 선택하여 활용할 수 있다. bc.로 표기된 인스턴스는 블록체인에 최적화된 인스턴스로 범용 모델인 T3타입, 인텔 제온 프로세서 기반의 범용 모델인 M5타입, 컴퓨팅 워크로드에 최적화된 C5 타입을 제공하고 있으며, bc.t3.small은 이중 가장 저렴한 타입으로 2개의 vCPU를 지원하며 2GB의 메모리를 탑재하고 있다. 인스턴스의 성능 및 용량이 높을수록 단위시간당 사용료 역시 높아진다[15].

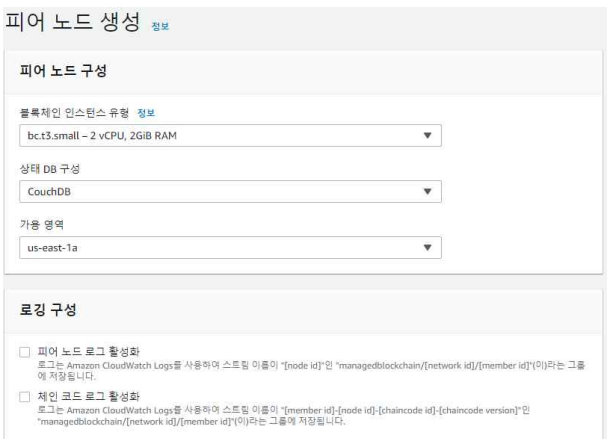

그림 13. AMB 피어 노드 생성
Fig. 13. AMB peer node creation

결과적으로 멤버당 하나의 피어노드를 구성하여 그림 14와 같은 하이퍼레저 패브릭 네트워크를 클릭 몇 번을 통해 매우 손쉽게 구성하였다.

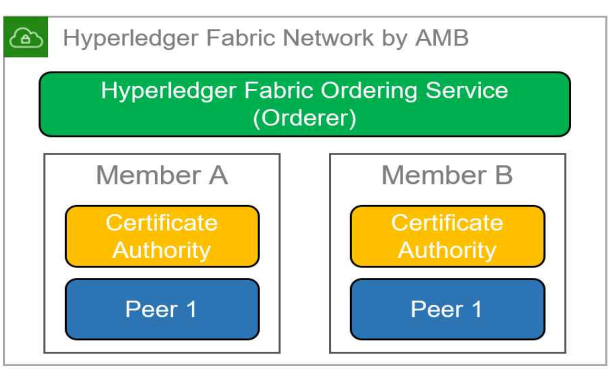

그림 14. AMB 기반의 Hyperledger Fabric 네트워크 구성
Fig. 14. AMB based Hyperledger fabric network configuration

4-2 AWS EC2 기반 Dapp 서버 구축

사용자에게 서비스 인터페이스를 제공하고, 사용자가 입력했거나 IoT 센서가 수집한 데이터를 데이터베이스 또는 블록체인 네트워크에 저장하는 역할을 수행하기 위해 AWS EC2를 활용하여 DApp 서버를 구축하였다.

1) DApp 서버용 AWS EC2 인스턴스 생성

먼저 DApp 서버용 EC2 인스턴스를 생성한다. AWS는 Linux, Linux2, Ubuntu, Windows, MacOS 등 다양한 Amazon Machine Image (AMI)를 제공한다[16]. 본 논문에서는 DApp 서버의 OS로 Linux2를 사용할 예정으로 인스턴스 생성 단계에서 프로젝트에 알맞은 AMI를 선택한다. 또한 인스턴스 타입, 보안 그룹, 인스턴스 세부정보, 스토리지, 태그를 설정하고 인스턴스를 생성한다. 해당 인스턴스는 DApp 클라이언트를 지원해야 하므로 보안 그룹 설정 시 반드시 HTTP, HTTPS 및 사용자 TCP/IP 포트를 추가한다. 그림 15는 이처럼 생성된 DApp 서버 인스턴스의 요약 정보를 보여준다. 생성된 EC2의 퍼블릭 DNS는 SSH로 EC2에 접속하거나 EC2가 서버로 구동시 DApp의 접속 주소가 된다.

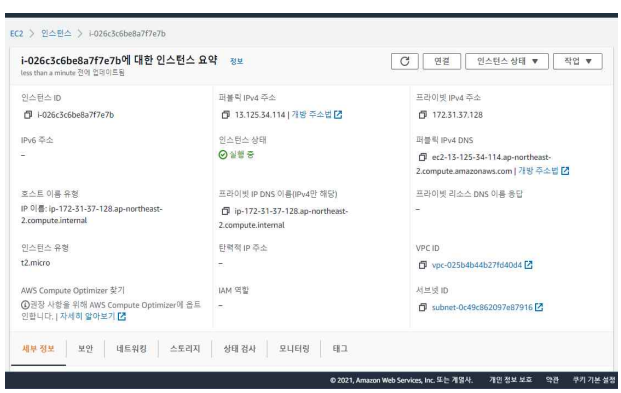

그림 15. DApp 서버 인스턴스 생성 결과
Fig. 15. DApp server instance creation result

2) DApp 서버 라이브러리 설치

DApp 서버 인스턴스가 생성되었다면 라이브러리 설치를 위해 해당 EC2에 SSH로 접속한다. 이는 인스턴스 생성시 제공된 AWS-keypair 파일인 .pem파일을 통해 허가된다.

본 논문에서는 백엔드를 위해 16.x 버전의 Node.js를 설치하였고 프론트 단을 위해 Vue.js 를 설치하고 Webpack을 포함한 Vue 프로젝트를 생성한다[17]. $vue init webpack frontend 설치 명령에서 vue-router를 yes로 설정한다.

또한 HTTP 통신, 세션관리 등을 위해 Express.js를 설치하고 $express backend --view=pug backend 명령을 통해 Express프로젝트를 생성한다[18].

다음으로 EC2의 퍼블릭 DNS로 프로젝트의 결과를 출력하기 위해 백 엔드 디렉토리 안에 있는 app.js를 수정하여 listen 포트 번호를 설정한다. 또한 node.js 수정내용을 반영하기 위해 package.json파일의 start의 값을 “nodemon app.js”로 수정한다. 설정이 모두 끝났다면 백 엔드 디렉토리에서 $npm start 명령어로 서버를 구동할 수 있다.

Vue.js와 Express.js는 각각 프론트 엔드와 백 엔드의 기능을 수행한다. 따라서 Vue.js에서 만들어진 프로젝트는 $npm run build를 통해 node.js파일로 변환되어 Express.js의 public경로에 생성되고 Express.js는 이 파일을 가지고 서버를 구동한다.

또한 프론트 엔드와 백 엔드 모두 동일한 모듈을 가지고 있어야 프론트 엔드에서 빌드를 했을 때 백 엔드에서도 정상적으로 구동이 가능하다. 본 프로젝트에서는 프론트에서 vue2-google-maps, axios, mysql module을 사용하기 때문에 연동을 위해 다음 명령어를 각각의 폴더에서 실행하여 모듈을 설치한다.

$sudo npm install vue2-google-maps
$sudo npm install axios --save
$sudo npm install mysql -save

4-3 배송 추적 서비스 백엔드 환경 구축

다음은 Node.js를 기반으로 구축한 배송 추적 서비스의 백엔드 환경 구축 결과를 보인다. 백엔드는 사용자의 요청사항, 입력사항 등을 처리하여 RDS 혹은 블록체인 네트워크에 이를 저장하거나 요청된 정보를 질의하여 제공한다. 이를 위해 먼저 다양한 서비스 요청을 처리하는 REST API 서버를 구축하고 실시간 데이터를 저장 및 관리하는 AWS RDS 기반 MariaDB를 구축하였다.

1) REST API 서버 구축

본 서버는 Node.js 기반으로 다음과 같은 기능들을 REST API 형태로 제공한다.

- blocklistener.js : 하이퍼레저 패브릭 네트워크 상에서 일어나는 블록 관련 이벤트 수신 기능 제공
- connection.js : 현재 사용자가 하이퍼레저 패브릭 네트워크의 인증기관에 등록된 사용자인지 확인하고, 네트워크와 클라이언트를 연결
- invoke.js : 블록체인 네트워크에 체인코드 제안을 전송
- query.js : 블록체인 네트워크의 데이터를 쿼리
- app.js : 배송서비스를 위한 사용자 등록 및 서비스 기능 제공 (하이퍼레저 패브릭 네트워크 인증기관에 사용자 등록, 생산자 및 사용자 코드 목록 반환, 특정 사용자 정보 및 주문 정보 질의, 배송 정보 및 IoT 센서노드로 부터 수집된 이상상황 데이터 업데이트)

2) AWS RDS 기반 MariaDB 데이터베이스 구축

AWS 데이터베이스 시스템 역시 다른 컴퓨팅 리소스와 동일하게 몇 번의 클릭만으로 구축이 가능하다. RDS는 AWS의 데이터베이스 시스템으로 EC2 생성과 동일하게 AWS 콘솔에서 생성한다. AWS 서비스 의 RDS 페이지에 접속하여 데이터베이스 생성을 클릭하면, 제원하는 데이터베이스 엔진을 선택할 수 있다[11]. 본 논문은 MariaDB를 선택하였으며, 데이터베이스 이름, 접속 계정 정보, 암호정보 등을 설정한다. 해당 데이터베이스 시스템이 배치될 VPC를 선택하고, 타 환경의 접속을 허용하기 위해 퍼블릭 액세스로 설정한다.

그림 16은 AWS의 RDS 인스턴스를 생성한 결과로, MariaDB 엔진을 사용하며, dapp-db 라는 이름으로 생성되었음을 확인할 수 있다. 외부 접속 허용을 위해 퍼블릭접근이 허용되었으며, 엔드 포인트 값을 통해 접속이 가능하다. 또한 생성할 때 입력한 마스터 사용자 이름 및 암호로 인증 후 접속할 수 있다.

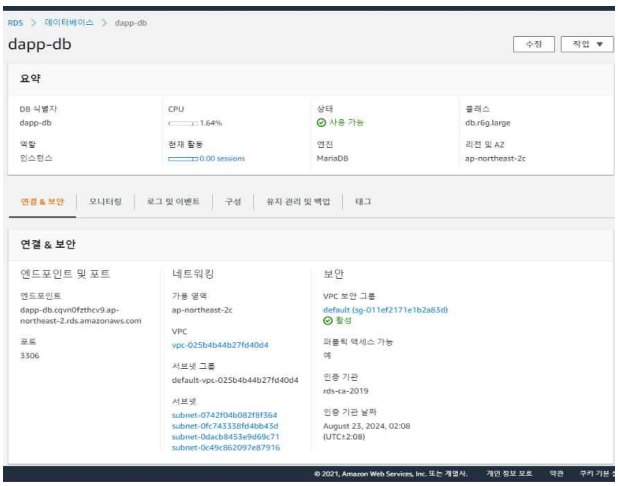

그림 16. 데이터베이스 시스템 생성 결과
Fig. 16. Database system creation result

3) 블록체인 기반 거래 처리를 위한 체인코드

하이퍼레저 패브릭 네트워크는 거래를 처리하기 위해 체인코드를 활용한다. 아래는 본 논문에서 정의한 체인코드를 설명하고 있다.

- queryByKey : 특정 데이터 쿼리를 위한 체인코드로 데이터를 쿼리하는데 필요한 키(ex:orderID)를 전달받아 블록체인 네트워크에서 해당하는 데이터를 가져온다.
- queryByString : 특정 데이터로 구분되어있는 데이터들을 모두 가져온다. (ex: 모든 주문 쿼리)
- createProducer : producerID를 키로 생산자 데이터를 생성한다. 만약 네트워크에 해당 키의 데이터가 존재하면 중복 에러를 반환한다.
- queryProducer/queryAllProducers : 특정 생산자 혹은 모든생산자 정보를 쿼리한다.
- createCourier : 생산자 데이터 생성과 마찬가지로 배송자의 CourierID를 통해 데이터를 생성한다.
- createOrder : 주문 데이터를 생성하며 성공시, 등록완료 상태정보와 등록 시간을 추가한다.
- updateOrder : 주문의 배송상태와, 해당 주문의 배송 담당자의 정보를 업데이트한다.
- createState : IoT데이터를 통한 물품의 상태 데이터를 생성한다. 이전 데이터를 덮어쓰지 않도록 배열 형태로 추가하고, 네트워크에 먼저 등록된 IoT 상태 데이터를 쿼리하여 추가할 IoT 상태 데이터와 병합하여 등록한다.

그림 17은 위에서 설명한 체인코드를 활용하여, 생산자 정보를 블록에 등록한 결과이다. REST API 형태로 호출된

createProducer 체인코드는 전달받은 생산자의 정보를 블록에 추가하며, 해당 트렌젝션은 7번 블록의 첫 번째 트렌젝션이며, 해당 요청이 성공적으로 등록되었음을 확인할 수 있다.

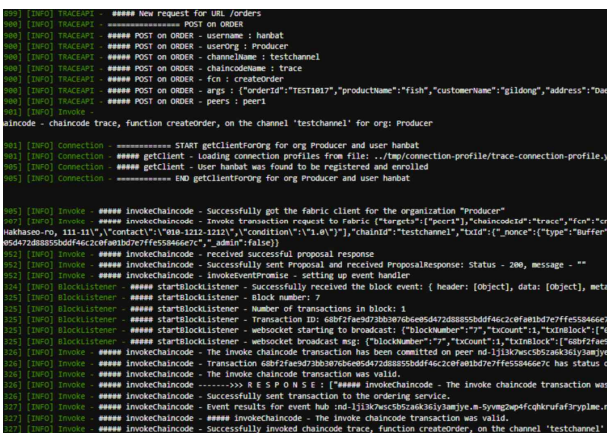

그림 17. 체인코드 기반 생산자 정보 업데이트 결과
Fig. 17. Chain code based creator information update result

4-4 라즈베리파이 기반 IoT 센서 노드 구현

본 논문에서는 상품의 배송 추적을 위해 라즈베리파이 기반의 IoT 센서노드를 구성하였다. 라즈베리파이 4B를 기반으로 온습도 센서 DHT22, 상품 위치를 확인하기 위한 GPS 센서 Neo-6M를 설치하였고 Python3.x 기반으로 코드를 작성하였다.

1) AWS 클라우드 기반 센서 데이터 수집

그림 18은 AWS 클라우드를 기반의 센서 데이터 수집 절차를 보인다. 라즈베리파이에서 수집되는 센서 데이터는 그 수집 간격이 매우 짧고 데이터 양이 많아 이를 직접적으로 AMB의 블록체인 네트워크에 적용하기는 효율적이지 않다. 따라서 본 논문에서는 센서노드에서 수집된 환경 데이터를 AWS의 데이터베이스 시스템인 RDS를 활용하여 1차적으로 저장한다. 이후 4-2장 및 4-3장에서 소개한 EC2 서버를 통해 중요 데이터를 AMB의 블록체인 네트워크에 저장한다.

그림 19는 제안하는 IoT 센서 노드의 환경정보 수집 및 전송 절차를 보이고 있다. 센서 노드는 온습도 센서를 통해 현재 상품이 노출되어 있는 온습도 정보를 수집하고 상품의 이상 상황을 판단하는 PBdata()함수를 통해 상품 상태값을 수집한다. 이후 GPS 센서를 통해 상품의 위치 정보를 수집한다. $GPGGA 값을 통해 실제 위치정보가 수집되지 않았다면 다시 센서 데이터 수집 단계를 반복한다[19]. 위치데이터가 유효하면 RDS의 MariaDB에서 현재 상품의 배송상태를 확인한다. 만약 해당 상품이 배송중이 아닐 경우, 실 데이터 전송은 하지 않고 대기하게 되며, 배송중일 경우 RDS에 배송상태, 날짜, 시간, 온도, 습도, 위치 데이터를 전송한다.

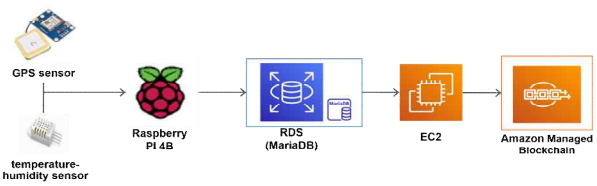

그림 18. 클라우드 기반 센서데이터 수집 절차
Fig. 18. Cloud based sensor data collection procedure

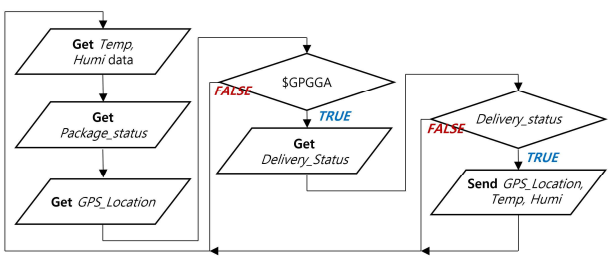

그림 19. 센서노드의 환경정보 수집 및 전송 절차
Fig. 19. Environmental information collecting and transmitting procedure of sensor node

2) 이상상황 판단 실험

앞서 설명한 것처럼, IoT 센서 노드는 상품의 이상 여부를 주기적으로 판단한다. 식품같이 신선도가 있는 상품의 경우 온도와 습도가 일정 범위내 유지가 된 상태에서 배송되야 하기 때문에 그 온도를 벗어난다면 배송 과정 중 상품의 변형이 생길 수 있고 이로 인해 상품의 문제가 발생했다고 볼 수 있다. 추가적으로 가속도 센서 등을 활용하면, 배송중 충격에 의한 파손 이벤트를 감지할 수도 있다.

그림 20은 상품의 이상상황을 판단하는 절차를 보인다. 본 실험에서는 간단히 온습도의 정상 범위를 초과했을 때 이상상황에 노출된 것으로 구성하였다. 배송되는 상품의 종류에 따라, 온습도의 정상 범위 및 이상 상황 노출 시간 등 여러 조건을 추가할 수 있다.

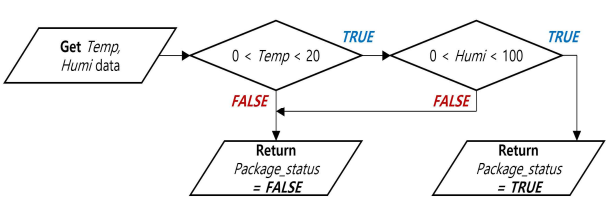

그림 20. 상품 이상상황 판단 절차
Fig. 20. Product abnormality determination procedure

그림 21은 본 연구에서 수행한 배송추적 실험을 보여준다. 배송할 제품을 식품으로 간주하고 온습도 유지를 위해 스티로폼으로 정사각형의 박스를 제작하였다. 보온력 증대를 위해 상자 내부에 일정 공간을 두고 2차 상자를 제작하여 부착했으며, 용이한 실험을 위해 투명한 아크릴로 내부를 볼 수 있도록 하였다. 상자의 상단부에는 라즈베리파이 센서 노드를 부착하였고, 온습도 센서는 상자 내부 뚜껑부분에 설치하였다. 그림 21의 오른쪽은 해당 상자를 들고 실제 실외로 이동하면

서, GPS 센서 데이터 수집을 테스트 하는 장면이다. 해당 실험 결과는 그림 26, 27과 같이 DApp을 통해 사용자가 배송 중 이상여부를 판단하고 추적 결과를 확인 할 수 있다.

그림 21. IoT 센서노드 기반 배송 추적 실험 환경
Fig. 21. IoT Sensor node based delivery tracking experimental environment

4-5 Vue.js 기반 배송 추적 서비스 DApp 개발

본 장에서는 Vue.js 기반으로 개발된 배송추적 서비스 DApp의 개발 결과물을 소개한다. Vue.js란 사용자 인터페이스를 만들기 위한 프로그레시브 프레임워크로, 뷰 레이어에 초점을 맞추어 다른 라이브러리나 타 프로젝트와 통합이 매우 쉽다. 작고 독립적이며 재사용할 수 있는 컴포넌트를 기반으로 구성되어 대규모 애플리케이션의 구축이 용이하며, 모든 유형의 애플리케이션 인터페이스를 컴포넌트 트리로 추상화할 수 있다[17].

애플리케이션 효율성을 위해 Vue Router를 기반으로 웹 페이지 간 이동을 제공한다. 라우팅은 SPA(Single Page Application) 방식을 사용하는데, 이는 제공할 페이지를 미리 받아 놓고 페이지 이동시에 클라이언트의 라우팅을 이용해 화면을 갱신하는 방식을 의미한다. 이를 활용하면 화면 간의 전환이 매끄러우며, 사용자 경험을 향상시킬 수 있다[20].

```
import Vue from 'vue'
import Router from 'vue-router'
import * as GgMap from 'vue2-google-maps'

//Main import
import main from '@/components/main'

//customer import
import CT_result from '@/components/customer/CT_result'
import GMap from '@/components/customer/GMap'
import QnA from '@/components/customer/Qna'

//producer import
import P_join from '@/components/producer/P_join'
import P_login from '@/components/producer/P_login'
import P_result from '@/components/producer/P_result'
import P_upload from '@/components/producer/P_upload'

//courier import
import C_join from '@/components/courier/C_join'
import C_login from '@/components/courier/C_login'
import C_result from '@/components/courier/C_result'
import C_upload from '@/components/courier/C_upload'
```

그림 22. 화면 라우팅을 위한 페이지 import 코드
Fig. 22. Page import code for screen routing

이와 같이 라우팅을 하기 위해서 먼저 라우팅을 할 페이지들을 import 시켜야 한다. 그림 22는 라우팅할 페이지를 import 한 코드로 어플리케이션에서 제공할 페이지를 미리 로드한다. 여기서는 사용자, 생산자, 배송자에게 제공할 페이지를 미리 로드하고 있다.

이와 같이 import된 페이지는 특정 이벤트가 발생되었을 때 함수를 실행하여, 컴포넌트 화면에 해당 페이지를 라우팅하여 보여주게 된다.

DApp은 AMB의 데이터를 질의하고 받기 위해 서버역할을 수행하는 EC2와 axios를 사용하여 통신한다. axios는 현재 뷰 커뮤니티에서 가장 많이 사용되는 HTTP 통신 라이브러리로 HTTP 통신에 대해 간단하고 직관적인 API를 제공한다. 그리고 API 형식이 다양하여 단순한 호출 이외에도 여러 설정 값을 추가하여 함께 호출할 수 있다. 아래는 DApp의 주요 기능의 구현 결과를 보여준다.

1) DApp의 로그인 및 회원가입

그림 23은 본 논문에서 구현한 DApp의 주요 기능 중 로그인 및 회원가입 기능을 보인다. 앞서 설명한 바와 같이, 생산자 및 배송자는 자신의 생산한 상품 또는 배달할 상품에 관한 관리 기능이 요구되므로, 회원가입 기능을 제공한다. ID, 이름, 비밀번호, 연락처를 입력받아 회원가입을 진행한다.

로그인을 할 때는 회원가입시 저장한 값과 비교 후 일치하면 로그인을 허용한다.

그림 23. 로그인 및 회원가입
Fig. 23. Login and sign-up

2) DApp의 상품 등록 및 주문 내역 확인

그림 24는 상품 등록 및 주문내역 확인 기능의 구현결과를 보인다. 생산자는 상품 등록을 위해, 상품명 및 해당 상품을 주문한 소비자 정보를 입력하며, 오른쪽 화면과 같이 주문 상세 내역을 확인할 수 있다.

그림 24. 상품 등록 및 주문내역 조회
Fig. 24. Product registration and order

3) DApp의 배송, 배상 조회

그림25는 배달상품 선택 및 배송중 상품 목록 기능을 보여준다. 배송자는 로그인시, 자신이 배달할 상품을 선택할 수 있다. 자신이 배달할 상품을 선택 후 배송출발 버튼을 누르면 IoT에 연결된 온습도 센서가 켜지면서 배송 상황 데이터 값이 Maria DB에 올라간다. 배송이 완료된 경우 특정 상품을 선택하고, 배송완료 버튼을 누르면, 해당 배송건에 대한 IoT 센서노드가 동작을 멈춘다.

그림 25. 배달상품 선택 및 배송 목록
Fig. 25. Delivery product select and delivery list

소비자는 배송업체가 자신에게 배송중인 상품의 배송 조회 기능을 통해 상품의 상태를 실시간으로 확인할 수 있다.

그림 26. 배송조회 및 배상문의 증명
Fig. 26. Delivery tracking and compensation procedure

만약 이상 여부가 true인 값이 있다면 상품의 품질에 악영향을 미칠 수 있는 상황이 발생한 것이다. 서비스는 배상문의 증명서를 통해 문제가 생긴 상품의 객관적 손해 및 파손 등의 책임소지 확인을 위한 각종 정보를 제공한다.

소비자가 주문번호로 상품조회를 하고 실시간 조회 기능을 선택하면, GPS 센서가 생성한 위치정보를 Maria DB에서 질의하여, 그림 27과 같이 구글 지도를 기반으로 현재 상품의 위치를 보여준다.

그림 27. 실시간 배송조회
Fig. 27. Real-time delivery tracking

V. 결 론

본 논문은 코로나19로 온라인 시장 및 배송서비스가 급증하면서 대두되고 있는 배송 과정 중 파손, 분실, 등의 문제에 주목하여, IoT, 클라우드 컴퓨팅 및 블록체인 기술을 활용한 신뢰적 배송추적 서비스 플랫폼을 제안하였다. 해당 플랫폼은 배송 과정에 라즈베리파이 기반의 IoT 센서를 도입하여, 배송받기 전까지 알 수 없었던 상품의 상태를 실시간으로 모니터링하며, 클라우드 기반의 데이터베이스를 통해 이를 저장/관리한다. 또한, 객관적이고 신뢰적인 배송 데이터 및 이벤트 처리를 위해 블록체인 기반의 스마트 컨트렉트를 통해 자동으로 블록체인 네트워크에 이벤트 정보를 기록하게 하였다.

이를 위해 본 논문 3장에서는 제안하는 서비스 플랫폼을 AWS의 대표적인 인스턴스를 기반으로 설계하였으며, 신뢰적인 배송과정의 추적을 위한 서비스 절차를 제안하였다. 또한 4장에서는 AMB 기반의 하이퍼레저 패브릭 네트워크 구성, AWS EC2 기반의 DApp 서버 및 배송추적 서비스 백엔드 환경 구축, 라즈베리파이기반 IoT 센서노드 구축, Vue 기반 DApp 구축 결과를 보였다.

제안하는 서비스를 기반으로 생산자는 자신이 발송한 물품을 Dapp을 통해 등록하여, 실시간, 신뢰적인 플랫폼을 기반으로 관리할 수 있으며, 배송자의 경우, 상품의 품질에 악영향을 미칠 수 있는 상황을 객관적 플랫폼을 통해 처리함으로써, 손해 및 파손 등의 과실책임 입증에 대한 객관적 정보를 활용할 수 있다. 소비자는 자신이 주문한 물품의 실시간 배송 상황 및 상태정보를 제공받고, 물품의 손실 발생 때, 배상이 용이하게 될 것이며, 결과적으로 배송과정의 신뢰도 및 만족도 향상으로 이어질 것으로 생각된다.

향후 의약품, 항공운송 등 배송 표준이 정립되어있는 분야의 요구사항 분석을 통해, 본 논문에서 제안한 배송추적 서비스 플랫폼을 고도화할 계획이다.

감사의 글

이 논문은 2022년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2020-0-00833, 5G 기반 지능형 IoT 트러스트 인에이블러 핵심기술 연구)

참고문헌

[1] B. S. Baek, “Daily online shopping...43% more than before COVID-19,” *ZDNet Korea*, [Internet]. Available: https://zdnet.co.kr/view/?no=20211012080514

[2] Y. J. Yoon, J. H. Lee, J.H. Oh and W. S. Rhee, “DApp design for supply tracking service based on AMB,” *in Proceeding of the 23th Conference on Electronics & Information Communications,* Daejeon, pp. 1-3, Dec. 2021.

[3] J. Y. Lee, “Trends and implications of blockchain technology,” *Science and technology policy institute, Trends and issues*, Vol. 34. PP. 1-21, July 2017.

[4] J. H. Park, “A Study on BaaS as a Key Engine for the Development and Application of Blockchain,” *Software policy & research institute, Monthly software oriented society*, July 2021.

[5] A. Long, D. Choi and J. Coffman, "Amazon Managed Blockchain for ePHI An Analysis of Hyperledger Fabric and Ethereum," *in Proceeding of the 2022 IEEE World AI IoT Congress(AIIoT)*, Seattle: WA, pp. 276-282, July 2022. https://doi.org/10.1109/AIIoT54504.2022.9817198

[6] J. Holbrook, “Deploying Your Blockchain on BaaS in Architecting Enterprise Blockchain Solutions,” *Wiley Data and Cybersecurity*, pp. 183-239, 2020.

[7] M. Onik, M. and Miraz, “Performance Analytical Comparison of Blockchain-as-a-Service (BaaS) Platforms,” *in Proceeding of the Emerging Technologies in Computing, iCETiC 2019*, London, pp. 3-18, Aug. 2019. https://doi.org/10.1007/978-3-030-23943-5_1

[8] AWS. Amazon Managed Blockchain [Internet]. Available: https://aws.amazon.com/ko/managed-blockchain/?nc2=h_ql_prod_bl_amb

[9] Y. Chen, J. Gu, S. Chen, S. Huang and X. S. Wang, “A Full-Spectrum Blockchain-as-a-Service for Business Collaboration,” *in Proceeding of the 2019 IEEE International Conference on Web Services (ICWS)*, Milan, pp. 219-223, Aug. 2019. https://doi.org/10.1109/ICWS.2019.00045

[10] AWS Partner network blog. Supply chain tracking and traceability with IoT-enabled blockchain on AWS [Internet]. Available: https://aws.amazon.com/blogs/apn/supply-chain-tracking-and-traceability-with-iot-enabled-blockchain-on-aws/

[11] AWS. Amazon RDS [Internet]. Available: https://aws.amazon.com/rds/

[12] J. H. Lee, J.H. Oh, Y. J. Yoon and W. S. Rhee, “Design of Supply Service System Architecture using Amazon Blockchain AMB,” *in Proceeding of the Symposium of the Korean Institute of communications and Information Sciences, Jeju, pp. 1428-1429, June 2021.*

[13] AWS Hyperledger fabric developer guide, get started creating a hyperledger fabric blockchain network using amazon managed blockchain [Internet]. Available:

https://docs.aws.amazon.com/managed-blockchain/latest/hyperledger-fabric-dev/managed-blockchain-get-started-tutorial.html

[14] J.H. Oh, J. H. Lee, Y. J. Yoon and W. S. Rhee, "Comparative Analysis of Amazon Blockchain AMB and IBM Hyperledger Fabric," *in Proceeding of the Symposium of the Korean Institute of communications and Information Sciences, Jeju, pp. 1444-1445, June 2021.*

[15] AWS. Amazon EC2 instance types [Internet]. Available: https://aws.amazon.com/ec2/instance-types

[16] AWS documentation. Amazon Machine Images (AMI) [Internet]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

[17] Vue.js. The progressive JavaScript framework [Internet]. Available: https://vuejs.org/

[18] Express. Express4.18.1, Fast, unopinionated, minimalist web framework for Node.js [Internet]. Available: https://expressjs.com/

[19] A. A. B. Ariffin, N. H. A. Aziz and K. A. Othman, "Implementation of GPS for location tracking," *in Proceeding of the 2011 IEEE Control and System Graduate Research Colloquium*, Shah Alam, pp. 77-81, June 2011. https://doi.org/10.1109/ICSGRC.2011.5991833

[20] MDN Web Docs Glossary. SPA (Single-page application) [Internet]. Available: https://developer.mozilla.org/en-US/docs/Glossary/SPA

최환석(Hoan-Suk Choi)

2009년 : 한밭대학교 멀티미디어공학과 (공학사)
2011년 : 한밭대학교 멀티미디어공학과 (공학석사)
2018년 : 한밭대학교 멀티미디어공학과 (공학박사)

2015년～현 재: 한국 ITU연구위원회 국제표준전문가
2018년～2020년: 한밭대학교 멀티미디어공학과 박사후연구원
2020년～2021년: 한국과학기술원 전기및전자공학부 위촉연구원
2021년: 주식회사 토브데이터 매니저
2021년～현 재: 카이스트-메가존클라우드 지능형 클라우드 융합기술 연구센터 매니저
※관심분야 : IoT, Social IoT, Semantic Processing, Trust management, Distributed ledger, Trust Chain, 개인정보보호, GDPR, Cloud Computing

오지훈(Ji-Hoon Oh)

2022년 : 한밭대학교 정보통신공학과 (공학사)

2020년～2021년: 한밭대학교 정보통신공학과 학부연구생
2021년～현 재: (주)솔탑 연구원
※관심분야 : Server Programing, Network Programing, VR/AR

이주형(Ju-Hyeong Lee)

2022년 : 한밭대학교 정보통신공학과 (공학사)
2022년～현 재 : 아주대학교 정보통신대학원 사이버보안학과 (석사과정)

2020년～2021년: 한밭대학교 정보통신공학과 학부연구생
2022년～현 재: 대한민국 육군 장교
※관심분야 : Blockchain, Ethereum, HyperLedger, Smart contract, IoT, AWS, Cloud Computing, Cyber Security, Blockchain Security, Cloud Computing Security

윤영주(Young-Joo Yoon)

2022년 : 한밭대학교 정보통신공학과 (공학사)

2020년～2021년: 한밭대학교 정보통신공학과 학부연구생
※관심분야 : Blockchain, Smart Contract, IoT, AWS, Docker, Vue, Data base

이우섭(Woo-Seop Rhee)

1983년 : 홍익대학교 (공학사)
1995년 : 충남대학교 (공학석사)
2003년 : 충남대학교 (공학박사)

1983년～2005년: 한국전자통신연구원 팀장/책임연구원
2005년～현 재: 한밭대학교 지능미디어공학과 교수
2006년～현 재: 한국ITU연구위원회 국제표준전문가
2012년～2013년: 프랑스 Institute TelecomSudParis 방문교수
2018년～2019년: 영국 Liverpool John Moores University 방문교수
※관심분야 : Semantic Processing, Trust Management, Trust chain, IoT, Social IoT