Connected type 분리형 플랫폼 상에서 차량 간의 실시간 정보공유 시스템 구현

이연재, 지유석*, 이창수**

㈜이화네트웍스

70003738@riwha.co.kr, jus2236@riwha.co.kr, cslee@riwha.co.kr

Implementation of real-time information sharing system between vehicles in a connected type separated platform

Lee Yeon Jae, Ji Yu Seok*, Lee Chang Su**

Riwha networks.

요 약

Connected Car 는 V2X(Vehicle to X)로 대변되는 기술들을 기반으로 차량과 차량, 차량과 교통인프라 등과 통신한다. 그리고 안전한 자율주행 또는 주행보조기능을 제공하거나, 차량 자체와 교통흐름 등에 대한 정보도 주고받는다. 본 논문에서는 고유한 WebIID 를 차량과 교통인프라에 부여한 Connected type 분리형 플랫폼에서 주행중인 차량들이 신원확인(IDP: Identification Process)[1]을 하고 자율주행에 필요한 정보를 실시간으로 공유하는, 차량(이동형 사물)간의 실시간 정보공유시스템을 구현하였다. 각 차량마다 포터블 클라우드 서버를 갖춘 차량(PV-C: Portable Cloud in Vehicle)들이 분리형 플랫폼에서 독립적인 네트워킹을 함으로써 플랫폼에서 발생하는 트래픽과 보안의 위험을 감소시키는 방법을 제안한다.

Ⅰ. 서 론

자율주행 방식에는 차량 자체로 자율주행 기능을 갖추는 형태(stand-alone type)와 주변 차량 및 교통 인프라와 협력(connected type)을 통한 방식이 있다[2]. 최근에는 두 type 간의 융합을 통해 자율주행기술들을 네트워크를 통해 연결하여 더욱 완성된 스마트 카를 지향하는 추세로, 많은 서비스 개발 업체들은 이러한 자율주행방식의 connected type 을 위한 구성으로 수많은 차량이 하나의 클라우드 서버와 연결된 구성을 예상하고 있다. 하지만 이러한 방식은 커넥티드 카와 클라우드 간의 수많은 데이터 전송이 일어나 데이터 트래픽 전송을 크게 증가시켜 응답 시간이 불필요하게 길어지고 계산 시간이 늘어나게 된다. 또한, 많은 커넥티드 카들이 클라우드에 등록되어 있을 경우에 서버가 해킹될 경우 많은 개인정보의 유출이 있을 수 있다. 따라서 실용적인 플랫폼을 구축하려면 결국 방대한 양의 데이터 처리를 수용할 수 있는 새로운 분산 네트워킹 및 시스템 아키텍처와 같은 기술이 필요하다. 본 논문에서는 이러한 문제를 해결하기 위해 차량 대 차량과 차량 대 교통인프라 통신을 분리한 Connected type 분리형 플랫폼을 구성하고 그 중 차량 대 차량간의 실시간 정보 공유 시스템을 구현하였다.

분리형 플랫폼에 적용된 차량(이동형 사물)을 포터블 클라우드 차량(PC-V)이라고 정의하였다. PC-V 는 교통 인프라를 통해 주행 중인 PC-V 들과 직접 접속하고, 실시간 주행정보를 공유함으로써 플랫폼에서 발생하는 트래픽의 위험을 감소시킨다.

Ⅱ. 본론

1. 시스템 구성도

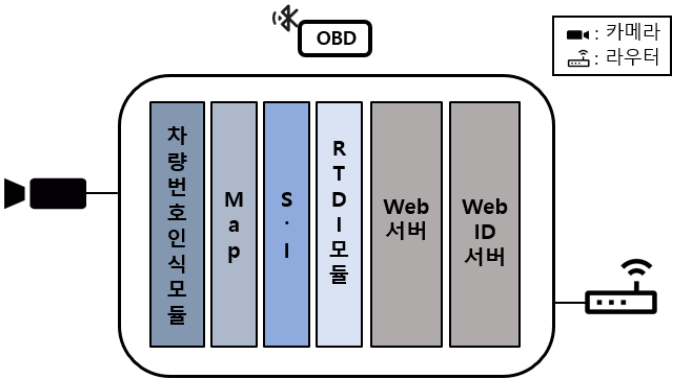

그림 1. 시스템 구성도

차량에는 그림 1 에서처럼 Web Server, WebID[3] Server, Service Interpreter, 네비게이션, RTDI(Real-Time Driving Information)모듈과 차량번호 인식 모듈을 갖춘 PC-V 용 시스템을 설치하였다. 그리고 PC-V 용 시스템은 LTE router, RTDI 와 연동되는 OBD-II Scanner, 차량번호 인식모듈과 연동되는 카메라를 연결하였다. 각 모듈의 역할은 다음과 같다. 앞차의 차량 번호를 인식하는 차량번호 인식 모듈, 네비게이션 map, 차량간 신원확인절차를 거쳐 통신을 하기 위한 S.I(Service Interpreter), OBD-II Scanner 를 통해 차량 주행정보를 실시간으로 받아오는 RTDI 모듈, 정보를 저장하기 위한 Web 서버, 차량 고유의 이름을 부여해 주는 WebID 서버가 있다. WebID server 는 SOLID(Social Linked Data)에서 제공하는 Solid server[4]을 이용하였다.

본 연구에서는 PC-V 용 시스템으로 Intel i5 CPU, SSD 256G, RAM 8G 사양의 laptop 컴퓨터를 사용하였다.

2. YOLOv5 를 이용한 차량번호 인식

차량번호 인식은 정확도를 높이기 위해 크게 번호판을 추출하는 부분과 추출한 번호판을 바탕으로 차량번호를 인식하는 부분으로 나누고 각각 데이터셋 구축, 모델 학습과 결과, 예측 순으로 진행한다.

먼저 번호판 추출은 클래스가 하나밖에 없기 때문에 데이터셋이 작아도 정확도가 높다. 따라서 데이터셋을 구축할 때 직접 촬영한 이미지 350 장과 데이터 증폭을 통한 이미지 350 장을 준비하고 준비한 이미지들의 라벨링을 진행한다. 라벨링은 머신러닝이나 딥러닝 모델 학습 전 이미지에 특정 값을 부여해 주는 작업으로 LabelImg 를 이용한다 [5]. 이렇게 구축한 데이터셋으로 YOLOv5s 모델을 이용해 학습을 진행한다 [6]. 모델 학습 결과 recall 값에 대응하는 precision 값의 평균값을 전체 클래스 개수로 나눈 값인 mAP 는 약 0.785 이고 이렇게 학습시킨 모델을 통해 번호판을 예측 후 추출한다.

추출된 번호판을 바탕으로 한 차량번호 인식은 클래스의 수가 매우 많아 클래스의 불균형이 일어날 수 있기 때문에 현실적으로 직접 촬영한 이미지는 데이터셋으로 사용하기 힘들다. 따라서 인조 번호판을 제작하여 데이터셋을 구축한다[7]. 인조 번호판은 균형 번호판과 일반 번호판 두가지로 균형 번호판은 한글 4 개, 숫자 4 개로 이루어져 있고 일반 번호판은 숫자 7 개, 한글 1 개로 이루어져 있다. 개수는 각각 2000 장, 1000 장으로 2:1 의 비율로 제작하여 데이터셋을 구축한다. 구축된 데이터셋을 바탕으로 위와 동일한 방법으로 YOLOv5s 모델을 이용하여 학습시킨 결과 mAP 는 약 0.95 이다. 최종적으로 아래 그림 2 와 같이 두 모델을 사용하여 번호판을 추출하고 추출된 번호판을 바탕으로 차량 번호를 예측하였다. 그림 2 에서의 정확도는 번호판 추출은 95%, 번호 인식은 평균 94.7%이다.

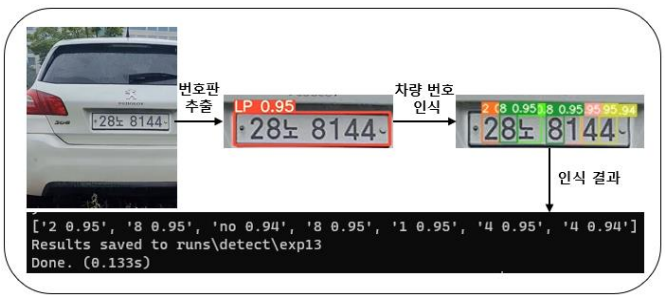

그림 2. 차량번호인식 과정

3. IDP (Identification Process)와 Update

IDP 는 차량 간의 신원 확인 절차로 뒷차와 앞차가 서로를 인식하는 과정이다[1]. 각 차량의 차량 구성도는 다음 그림 3 과 같다.

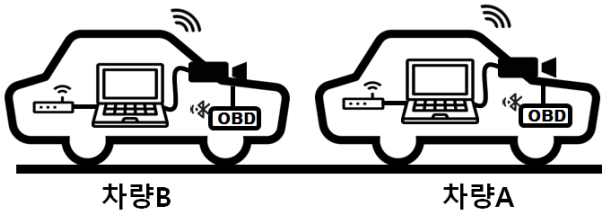

그림 3. 차량 구성도

차량A는 앞차, 차량B는 뒷차이고 각 차량에 대한 profile은 다음 표1과 같다.

차량	Profile
차량 A 28 노 8114	-WebID: https://28no8114.cns-link.net:8443/profile/card#me -WebServer 주소: http://28no8114.cns-link.net:3000 -고정 IP: 223.171.54.216 -앞 차의 web 서버 주소
차량 B 07 로 5456	-WebID: https://07ro5456.cns-link.net:8443/profile/card#me -WebServer 주소: http://07ro5456.cns-link.net:3000 -고정 IP: 223.171.55.16 -앞 차의 web 서버 주소

표 1. 차량 정보

표 1 에서처럼 Profile 에는 WebID, Web 서버 주소, IP 주소, 앞 차량 Web 서버 주소가 기록되어 있다. 이러한 Profile 은 각 사물의 기본 신원 정보 BII(Basic Identity Information)이다[1]. 표 1 에서처럼 각 차량은 WebID 를 이용하여 자신과 다른 차량을 구분해 줄 수 있는 <https://차량번호.cns-link.net:8443/profile/card#me> 형식의 고유한 이름을 가진다. 위와 같은 구성으로 차량 B 가 차량 A 에 IDP 를 하면 차량 B 는 계속해서 차량 A 의 RTDI 를 실시간으로 Update 한다. IDP 와 Update 의 전체 과정은 그림 5 와 같다.

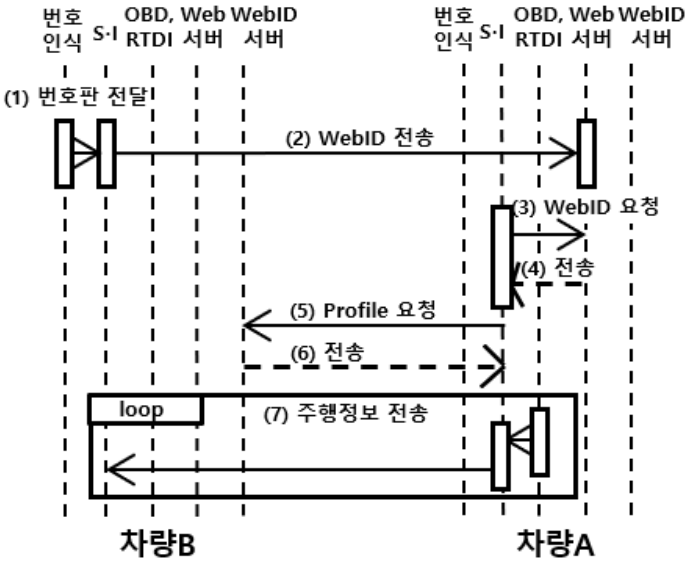

그림 5. IDP 와 Update 과정

그림 5 에서처럼 차량 B 가 차량 A 의 번호판을 인식하면 (1) 차량 B 의 번호인식 모듈이 S.I 에 번호판을 전달한다. (2) S.I 가 인식된 차량번호를 통해 웹 서버 도메인을 조합한 후 차량 A 의 Web 서버에 차량 B 의 WebID 를 전송한다. (3) 차량 A 의 S.I 가 Web 서버에 저장된 차량 B 의 WebID 를 요청하면 (4) Web 서버가 WebID 를 전송한다. (5) 확인한 WebID 를 통해 S.I 가 차량 B 의 WebID 서버에 profile 을 요청하면 (6) WebID 서버가 profile 을 전송한다. (7) 차량 B 의 profile 을 통해 차량 A 가 차량 B 의 S.I 와 연결한 후 OBD 를 통해 만들어진 RTDI 를 전달받아 전송한다.

4. 결과

IDP 는 그림 5 의 (1)~(6)번까지 걸리는 시간을 측정하고 Update 는 (7)번 과정을 측정한다. 측정하기 전 시간서버는 차량 B 를 기준으로 하였고 오차범위는 2~9ms 내외다. 테스트는 오송시내에서 이동하면서 측정하였고 이동하기 전 라우터 속도에 따른 IDP 측정결과는 다음 표 2 와 같다.

횟수	이동 전 라우터 속도	시간(ms)
IDP1	앞차: 5.7Mbps 뒷차: 1.2Mbps	931ms
IDP2	앞차: 6.7Mbps 뒷차: 5.9Mbps	729ms
IDP3	앞차: 5.6Mbps 뒷차: 7.9Mbps	759ms

표 2. IDP 측정결과

표 2 에서처럼 평균 806ms 가 걸리고 라우터 성능에 따라 시간도 상대적으로 오래 걸린다. IDP 이후 Update 는 1 초마다 주행정보를 전송하는 것으로 하였고 각 IDP 별 Update 측정결과는 다음 그림 6 과 같다.

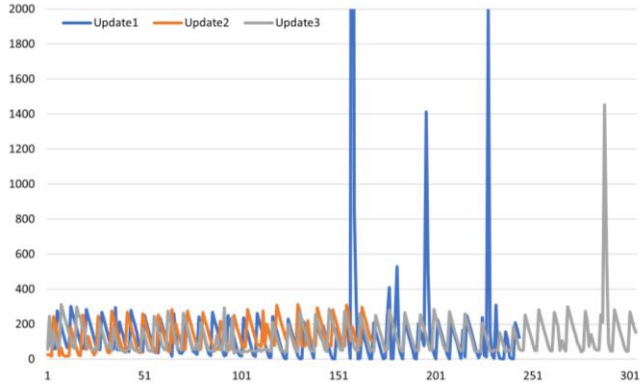

그림 6. Update 측정결과

그림 6 에서처럼 Update1 은 245 번, Update2 는 169 번, Update3 는 305 번 측정하였고 각각 평균 159ms, 135ms, 126ms 로 측정되었다. 위와 같은 데이터들은 LTE 라우터의 성능에 따라 매우 달라지고 성능이 좋은 라우터일수록 시간은 더욱 단축될 것으로 보인다.

Ⅲ. 결론

본 논문에서는 각각의 PC-V가 독립된 네트워킹을 통해 차량 간의 정보 공유 서비스를 직접 처리하는 방식을 제안한다. PC-V가 앞 차량을 인식하고 IDP를 완료하는 지연시간은 3GPP release 15 eV2X의 TR22.886에서 요구하는 각 차량의 어플리케이션 서버와의 상호인식 지연시간 1초에 근접하고, 차량이 앞 차량의 RTDI를 수신하는 지연시간은 TR22.886에서 요구하는 V2I, V2N에서의 지연시간 100ms 이내 근접함으로써 Connected type 분리형 플랫폼에서 이동형 사물 간의 실시간 정보공유 시스템이 3GPP release 15 eV2X에서 요구하는 수준을 맞출 수 있는 가능성을 본 연구에서 확인하여 보았다. 현재 전체 속도를 향상시키기 위해 LTE라우터의 성능 별 IDP, Update 속도를 측정하고 그에 따른 성능 향상 연구를 진행 중에 있다.

ACKNOWLEDGMENT

본 연구는 중소기업기술정보진흥원의 프로젝트 “자율주행을 위한 connected type 분리형 플랫폼 및 커넥티드카 IoT 디바이스 개발(S2883964)”의 지원을 받았습니다.

참 고 문 헌

[1] Chang-Su LEE, Sweung-Won CHEUNG, Seong-Soon, JOO, Hyun-Kook, KAHNG, “Design and Implementation of Autonomous Collaboration System of Smart Things using accumulated Experience knowledge,” International Conference on Advanced Communication Technology(ICACT) ISBN:979-11-88428-03-8, pp305-313, 2019.

[2]장필성, 백서인, 최병삼 (2018). 자율주행차 사업화의 쟁점과 정책 과제. 동향과 이슈(49), 1-31

[3] Tim Berners-Lee, Henry Story, Andrei Sambra, https://www.w3.org/2005/Incubator/webid/spec/identity/

[4] Tim Berners-Lee, Christian Smith, Sarven Capadisli, Dmitri Zagidulin, Nicola Greco and Ruben Verborgh, ”node-solid-server”, (https://github.com/solid/solid)

[5] Tzutalin, “LabelImg”, 2015 (https://github.com/tzutalin/labelImg)

[6] Jocher, G., Nishimura, K., Mineeva, T., Vilariño, R., “YOLOv5”,(https://github.com/ultralytics/yolov5).

[7] 이유진, 김상준, 박경무, 박구만 (2020). 3 차원 회전을 이용한 인조 번호판 생성기의 번호판 인식 성능 비교. 한국방송 미디어공학회 학술발표대회 논문집, 141-144