RSU내 컨테이너를 이용한 워크로드 처리 기법

Workload Processing Method using Containers in RSU

정진원[1], 이재학[1], 길준민[2], 유헌창[1*]

JinWon Jeong, JaeHak Lee, JoonMin Gil, HeonChang Yu

[1](02841) 서울특별시 성북구 안암로 145, 고려대학교 정보대학 컴퓨터학과

[2](38430) 경북 경산시 하양읍 하양로 13-13 대구가톨릭대학교 컴퓨터소프트웨어학부

{jin4812, smreodmlvl}@korea.ac.kr[1], jmgil@cu.ac.kr[2], yuhc@korea.ac.kr[1]

요 약

커넥티드 카 환경에서 모바일 엣지 서버의 작업이 보장되지 못할 경우, RSU로 워크로드를 오프로딩 하여 지속적으로 실시간 센서 데이터 처리가 유지되어야 한다. 하지만 RSU 내에서 효율적인 자원 관리가 이루어지지 않는다면 긴급한 정보를 수신하지 못해 차량의 안전이 보장되지 못할 수 있다. 따라서 본 논문에서는 RSU의 센서 데이터 처리에 필요한 자원 격리 및 데이터 처리 성능 보장을 위한 연구를 진행함으로써 RSU 서비스를 보다 안정적으로 제공할 수 있는 방안을 제안한다.

키워드: RSU(Road Side Unit), OBU(On Board Unit), MEU(Mobile Edge Unit), 컨테이너(Docker Container)

1. 서론

RSU(Road Side Unit) 장비는 급증하는 차량 사고 및 사용자 중심의 차세대 교통안전 서비스 제공을 목적으로 주행 중 또는 정차 중에 있는 차량 단말기(OBU, On Board Unit)와 상호 통신을 통해 안정적인 서비스를 고객에게 맞춤 제공하기 위한 RF(Radio Frequency) 기지국 장치이다. RSU는 도로 당국 또는 제 3 자(예: 주차 시설 운영자, 주유소 등)에 의해 고정된 위치에 배치되어 주로 백홀 및 로컬 관리 서비스를 제공하며 자신의 위치 정보를 브로드캐스팅 한다. 이들 RSU의 위치는 통신 범위 내에 있는 모든 차량에게 전달되어 이용이 가능하다.

직선도로에서 OBU와 RSU 간 수백 ms의 지연은 서비스 측면에서 큰 고려가 되지 않을 수 있으나, 교차로에서는 상기 두 디바이스 간 100ms의 서비스 지연으로 인해 교통사고가 발생할 수 있다[1]. RSU는 데이터 센터에 비해 컴퓨팅 자원이 열악하기 때문에 RSU로 대량의 데이터가 들어와 내부에서 일처리가 조금이라도 지연된다면, OBU가 주변 차량사고와 같은 유고 정보(REI, Road Event Information)를 포함하는 차량의 안전이 보장되어야 할 긴급서비스를 수신하지 못함으로 인해 위험한 상황에 직면할 수 있다[2]. 또한 데이터들에 대한 네트워크 I/O 및 CPU 작업이 자주 일어남으로 효율적인 RSU 자원 관리가 필요하다. 따라서 본 논문에서는 컨테이너(Docker Container)를 이용하여 자원이 격리된 환경을 구성한 후, RSU내 워크로드를 처리하여 RSU 서비스를 보다 안정적으로 제공할 수 있는 방안을 제안한다. 이를 바탕으로 RSU에서 컨테이너의 이용 여부에 따른 워크로드 처리 수행시간을 측정하여 그 결과를 비교 및 분석한다.

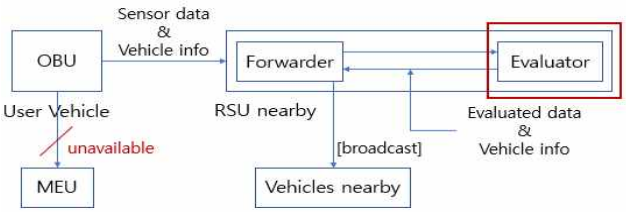
2. 관련 연구

기존 연구에서는 커넥티드 카를 위한 안전정보 및 사고방지 등의 서비스 제공을 위해 빠른 응답시간, 적은 네트워크 트래픽 지원이 가능하도록 개발된

* 교신저자

스마트 폰 기반 엣지 서버에서의 작업이 보장되지 못할 경우, RSU로 워크로드를 오프로딩 하여 지속적으로 실시간 센서 데이터 처리가 유지되도록 하기 위한 연구를 진행하였다. 또한 차량 내 부착된 OBU와 스마트 폰에 엣지 서버 기능이 탑재된 모바일 엣지 유닛(MEU, Mobile Edge Unit)간의 통신이 불가능한 상황을 나타내기 위해 (그림 1)과 같이 환경을 구성하였다. RSU가 MEU를 대신해서 OBU로부터 차량의 정보와 센서 데이터를 수신하면 내부 Forwarder를 통해 데이터가 Evaluator로 전달된다. Evaluator에서는 수신된 데이터를 통해 요청받은 차량의 상태 및 사고 여부를 판단하고 Forwarder에게 그 결과를 보고한 후, 근처에 있는 다른 OBU로 브로드캐스팅 한다.

하지만 기존 연구에서 제안된 RSU는 Evaluator에서 모든 워크로드를 혼자 처리하기 때문에 시스템 자원에 과부하가 발생한다면 워크로드 처리 지연시간이 커지는 문제점이 발생할 수 있다. 따라서 하나의 컴포넌트만을 두는 대신 RSU 매니저를 배치하여 실시간으로 데이터가 들어올 때마다 컨테이너를 생성한 후, 자원이 격리된 환경에서 워크로드를 처리하는 기법을 제안한다.



(그림 1) 기존 RSU 구성도

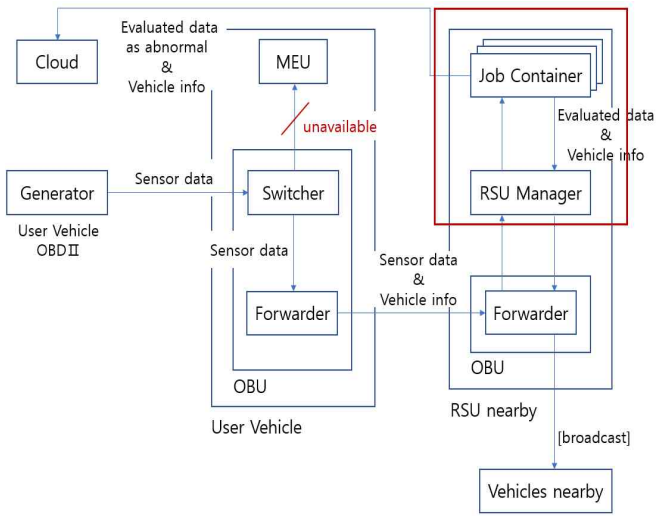
3. 시스템 구성도

3.1 MEU 구성도

본 연구에서 OBU는 Switcher와 Forwarder로 구성되어 있으며, Switcher는 OBD2Ⅱ(On Board Diagnostics)의 역할을 수행하는 Generator와 커넥티드 카 센서 데이터에 대한 통신을 수행하고, MEU와의 데이터 송수신을 담당한다. MEU에서 센서 데이터를 수신하여 정상 및 비정상 값 또는 사고 데이터가 탐지되면, 분석된 데이터를 다시 Switcher에게 송신한다. Forwarder는 표준 입출력 통신으로 Switcher로부터 해당 데이터를 전달받고, 단거리 전용 통신(DSRC, Dedicated Short-Range Communication) 유닛에게 차량의 정보와 데이터를 브로드캐스팅 함으로써 사고알림 정보를 전달한다.

3.2 제안하는 RSU 구성도

본 절에서는 OBU와 MEU간의 통신이 불가능 할 때의 상황을 다룬다. MEU에서 실시간 센서 데이터 처리가 불가능 하다고 판단될 경우, Switcher는 Generator로부터 받은 데이터를 Forwarder에게 바로 전달하고, Forwarder는 차량의 정보와 센서 데이터를 근처 RSU에게 송신한다. RSU에서는 OBU로부터 수신된 센서 데이터 처리에 필요한 자원 격리 및 데이터 처리 성능 보장을 위해 컨테이너를 이용하여 워크로드 처리를 수행하도록 한다. 컨테이너는 운영체제의 자원을 공유하기 때문에 실행 시 CPU, 네트워크, 메모리를 필요만큼만 자원을 할당해주며, 다른 컨테이너들이 접근하지 못하도록 격리한다. 즉, 하나의 OS에서 격리된 환경으로 독립된 OS가 운영되는 것처럼 보이게 해준다. 따라서 컨테이너를 이용함으로써 서버 전체 자원을 효율적으로 사용하여 보다 안정적인 RSU 서비스를 기대할 수 있다. (그림 2)에서와 같이 RSU 매니저가 Forwarder로부터 차량의 정보와 센서 데이터를 수신하면 그에 해당하는 컨테이너를 생성한다. 컨테이너 안에서 워크로드를 처리하여 해당 차량에 대한 상태 및 사고 여부가 판단되면 RSU 매니저에게 그 정보를 전달하고, 차량의 상태가 비정상이거나 사고가 발생한 것으로 판단되면 클라우드 서버에게도 전달한다. 그 후 컨테이너는 소멸된다.



(그림 2) 제안하는 RSU 구성도

4. 실험

4.1 실험 환경 및 수행 방법

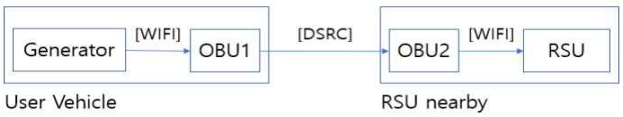
본 실험에서는 RSU에서 컨테이너의 이용 여부에 따라 MEU의 기능이 상실되었을 시 오프로딩 된 워크로드를 처리하는데 걸린 수행 시간의 증가율을 비교하였다. 본 실험을 위해 (그림 3)과 같이 커넥티드카 환경에서 발생하는 센서 데이터들을 실시간으로 통신하여 처리할 수 있는 환경을 구성한다. Generator는 실제 차량의 데이터를 이용하여 OBU로 차량 데이터를 보내주는 역할을 하는 프로그램이며, 수집한 차량 데이터는 csv 파일로 구성되어 있고 C# 언어로 개발되었다. RSU와 OBU의 환경은 <표 1>과 같다.

<표 1> RSU와 OBU의 환경

	RSU	OBU
MODEL	OptiPlex 3070	ETF-DO-02
CPU	i7-9700 CPU @ 3.00GHz x 8	Quad ARM Cortex-A53
RAM	32GB	4GB
OS	Ubuntu 18.04.4 LTS	Linux 4.9.58

먼저 Generator에서 차량의 센서 데이터가 생성되면 웹 소켓통신으로 센서 데이터를 OBU1로 전송한다. OBU1은 MEU와 통신이 불가능한 상태이므로 OBU2에게 DSRC 통신을 이용해 차량의 정보와 센서 데이터를 전달하고, OBU2가 이 정보와 데이터를 웹 소켓통신으로 RSU에게 최종적으로 전달한다. 이때 RSU에서 시스템 자원에 과부하가 발생하는 상황을 연출하기 위해 대량의 유고 정보가 RSU로 송신되는 것을 각 실험마다 구분하여 I/O와 메모리, 그리고 CPU 자원 과부하로 나타내었다. 각 자원 영역에 부하 작업을 수행하는 애플리케이션으로는 리눅스 과부하 테스트 프로그램인 stress tool[3]을 이용하였다.

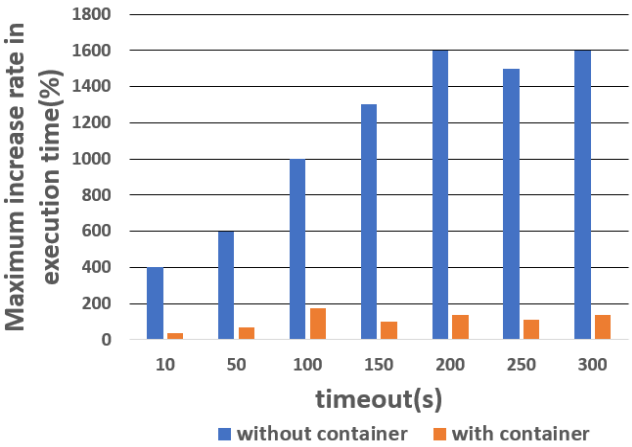
실험을 진행하기 위해 RSU가 OBU로부터 차량의 정보와 센서 데이터를 수신하여 워크로드를 처리함과 동시에 각 실험마다 구분된 해당 자원에 과부하를 준다. 부하를 주기 전에는 일정한 시간으로 워크로드를 처리하다가 부하가 시작되면 수행시간이 크게 변동되는 현상이 발생한다. 부하를 주는 시간(timeout)은 10초에서 시작하여 300초까지 주었을 때, 각 실험에서 도출된 워크로드 처리 최소 수행시간과 최대 수행시간을 측정하여 그 증가율을 그래프로 나타낸다. 이 과정을 RSU에서 컨테이너의 이용 여부에 따라 수행하고 도출된 결과 값을 비교한다.



(그림 3) RSU와 OBU 간 실험 환경

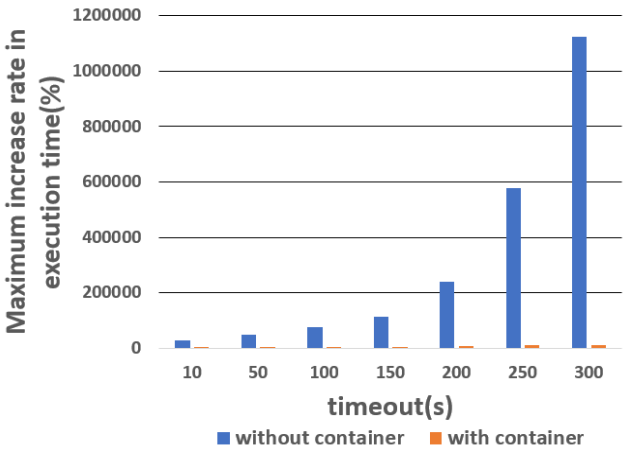
4.2 실험 결과 및 분석

(그림 4)는 1000개의 스레드로 I/O 영역에 부하를 주었을 때 도출된 워크로드 처리 수행시간의 최대 증가율을 비교한 그래프이다. 컨테이너를 이용하였을 때의 경우가 이용하지 않았을 때보다 워크로드 처리 수행시간의 최대 증가율이 비교적 매우 작게 나타나며 이는 컨테이너에서 제공하는 자원 격리성이 반영된 결과로 볼 수 있다.



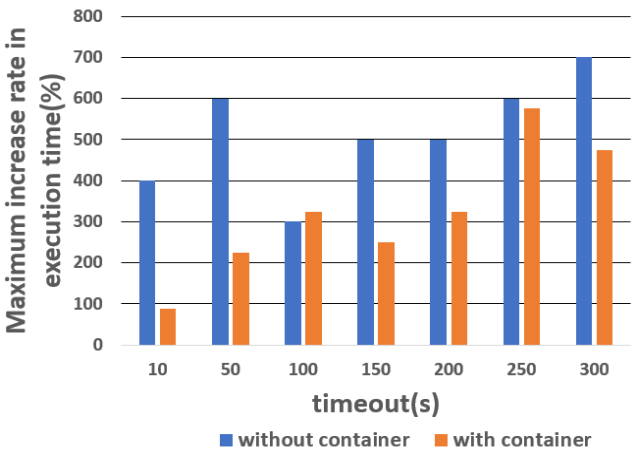
(그림 4) I/O 영역 부하: 수행시간 최대 증가율

(그림 5)는 200개의 스레드로 메모리 영역에 부하를 주었을 때 도출된 워크로드 처리 수행시간의 최대 증가율을 비교한 그래프이다. I/O 영역에 부하를 주었을 때와 유사하게 컨테이너를 이용하였을 때의 경우가 워크로드 처리 수행시간의 최대 증가율이 비교적 매우 작게 나타나는 모습을 확인할 수 있다.



(그림 5) 메모리 영역 부하: 수행시간 최대 증가율

(그림 6)은 1000개의 스레드로 CPU 영역에 부하를 주었을 때 도출된 워크로드 처리 수행시간의 최대 증가율을 비교한 그래프이다. 위의 두 가지 실험 결과와는 다르게 컨테이너를 이용했을 때의 경우, 100초 지점에서 워크로드 처리 수행시간의 최대 증가율이 조금 더 크게 나타났으며, 나머지 지점에서는 두 결과값의 차이가 위 두 실험에 비해 비교적 많이 크지 않은 모습을 확인할 수 있다.



(그림 6) CPU 영역 부하: 수행시간 최대 증가율

5. 결론

본 논문에서는 다양한 교통 정보 서비스를 제공하는 RSU에서 컨테이너를 이용하여 발생되는 워크로드 유형에 따른 성능 보장을 위해 자원 격리성을 높여, OBU로부터 받은 차량 및 센서 정보를 효율적으로 처리하는 방안을 제안하였다. 본 실험 중 CPU 영역에 부하를 주는 실험에서 컨테이너가 제공하는 자원 격리성이 다른 두 실험에 비해 두드러지게 나타나진 않았지만, 컨테이너를 이용함으로써 워크로드 처리 수행시간의 증가율이 컨테이너를 이용하지 않았을 때보다 비교적 작다는 것을 보여주었다.

본 실험을 통해 RSU에서 시스템 자원에 과부하가 발생할 경우 컨테이너를 이용하여 보다 안정적으로 워크로드를 처리할 수 있었지만, 과부하가 발생하지 않는 상황에서 워크로드 처리 수행시간만을 비교하였을 때는 컨테이너를 이용하지 않았을 때의 경우가 더 빠른 결과로 나타났다. 그 원인으로는 차량의 정보 및 센서 데이터가 들어올 때마다 컨테이너가 생성되기 때문에 그 시점이 영향을 주어 워크로드 처리 수행시간이 비교적 느리게 나타난 것으로 보인다. 따라서 컨테이너의 생성 시점에 초점을 두어 속도에 관한 부분을 더 보완할 예정이다.

본 연구에 대한 향후 연구로는 RSU에서 우선순위 기반으로 센서 데이터를 처리하는 방향을 생각할 수 있다. RSU로 들어오는 무분별한 데이터 중 가장 긴급한 정보를 OBU에게 전달해야 하는 상황이 발생할 수 있으므로, 컨테이너의 워크로드에 대하여 작업 우선순위를 고려해 우선적으로 처리해야 하는 작업을 식별하여 처리할 수 있는 기법을 연구할 계획이다.

Acknowledgement

I “본 연구는 과학기술정보통신부 및 정보통신기획 평가원의 대학ICT연구 센터지원사업의 연구결과로 수행되었음” (IITP-2018-0-01405)

II 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획 평가원의 지원을 받아 수행된 연구임 (No.2018-0-00480)

참고문헌

[1] 조순기, 정회빈, 이석준, “국내 Cooperative ITS 도입을 위한 서비스 규격서”, 국토교통부 첨단도로환경과, Ver 1.1, 2013.

[2] Seif Ben Chaabene et al., “A Roadside Unit Placement Scheme for Vehicular Ad-hoc Networks”, 2019.

[3] https://linux.die.net/man/1/stress