논문 2024-19-26

# 디바이스리스 컴퓨팅을 위한 MEC기반 대규모 엣지 디바이스 모니터링 기술 연구

## (MEC-Based Massive Edge Device Monitoring Techniques for Deviceless Computing)

**전 인 걸*, 석 종 수**
(In-geol Chun, Jong-soo Seok)

Abstract : As computing technology advances, many services, including AI, that previously operated in the cloud will become usable on devices that users carry. The emergence of ultra-high-speed mobile networks like 5G dramatically increases the utility of numerous devices in the real world. In the future, with technologies like deviceless computing, the range of applications will diversify even further, and demand will continue to grow. Consequently, the importance of technology for monitoring vast amounts of device information and deploying AI services tailored to the functions and performance of each device is becoming increasingly evident. Therefore, this paper proposes a large-scale edge device monitoring technique necessary to leverage simple sensors and low-spec, low-resource devices in conjunction with Multi-access Edge Computing (MEC) to provide various AI functionalities.

Keywords : Massive Device Monitoring, Edge Device Manager, Deviceless Computing, Multi-access Edge Computing, Computing Continuum

## Ⅰ. 서 론

클라우드 컴퓨팅은 현대의 작업 및 생활 환경에서 발생하는 대용량 데이터를 안전하고 경제적으로 관리할 수 있는 최적의 솔루션으로 평가된다. 그러나 컴퓨팅 기술의 발전과 함께 다양한 전통 산업 분야에 대한 컴퓨팅 기술의 도입이 급속히 진행되고 있으며, 하드웨어 기술의 발전으로 인해 산업 분야별로 활용되는 디바이스의 수가 급증하고 있다. 이러한 상황은 클라우드를 통한 대용량 데이터의 저장 및 관리만으로는 해결하기 어려운 여러 문제를 초래하고 있다. 특히, 5G로 대표되는 무선 통신망의 발전은 단순한 모바일 단말기를 넘어 현실 세계의 수많은 엣지 디바이스에 무선 통신의 활용을 촉진하고 있다. 최근 ChatGPT와 같은 인공지능 기술의 등장으로 일반 대중이 인공지능 기술을 직접 체험할 수 있는 기회가 확대됨에 따라, 실생활에서 인공지능 기술 활용에 대한 요구가 증가하고 있다. 이에 따라, MEC (Multi-access Edge Computing)와 같은 엣지 디바이스와 근접한 장치를 활용하여 수많은 디바이스의 정보를 실시간으로 관리하고, 각 디바이스의 기능 및 성능에 적합한 AI 서비스를 제공하기 위한 대규모 엣지 디바이스 모니터링 및 관리 기술의 중요성이 부각되고 있다 [1]. 특히, 엣지 디바이스가 동작하는 실세계의 다양한 주변 환경과 사람들의 행동 방식, 그리고 여러 종류의 디바이스에 대한 분석을 통해 의미 있는 데이터를 수집·생성하여 최적의 운영 방법을 결정하는 것은 매우 중요하다. 따라서 본 연구는 자율주행 자동차, 스마트 시티, 스마트 홈 등 다양한 환경에서 수백 개에서 수만 개의 디바이스가 활용되는 상황을 고려하여, 이러한 디바이스의 정보를 실시간으로 모니터링하고 통합적으로 관리하기 위한 기법을 제안하고자 한다. 본 논문의 2장에서는 연구의 배경과 제안하는 연구와 연관되는 기술에 대해 설명한다. 3장에서는 대규모 엣지 디바이스 관리기의 구체적인 설계안을 제시하고, 4장에서는 실제 시스템을 구현하여 제안한 기술의 활용가능성을 검증하였다. 마지막으로 5장에서는 제안한 연구결과에 대한 요약과 향후 연구계획을 기술한다.

## Ⅱ. 연구 배경 및 관련 연구

엣지 디바이스가 다양한 시장 및 산업으로 확대됨에 따라 기술은 수많은 디바이스와 컴퓨팅 시스템간 연결되고 실시간 처리 및 효율적인 에너지 사용에 대한 수요가 지속적으로 증가하고 있다. 특히 모바일 및 IoT 장치의 수와 유형이 급증함에 따라 디바이스로부터 주변 환경에 대한 막대한 양의 데이터가 생성되고 있어 데이터 전송 지연시간 감소, 개

*Corresponding Author (igchun@etri.re.kr)
Received: Aug. 21, 2024, Revised: Sep. 9, 2024, Accepted: Oct. 4, 2024.
I. G. Chun: ETRI (Principal Researcher)
J. S. Seok: ETRI (Senior Researcher)
※ 이 논문은 2024년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00546, 열린혁신디지털오픈랩).

ⓒ IEMEK J. Embed. Sys. Appl. 2024 Oct. 19(5) 211-218
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2024.19.5.211

인정보보호 강화, 클라우드 리소스 및 대역폭 비용 절감 등의 요구가 급증하고 있다. 그러나 기존 연구는 대부분 센서와 같은 엣지 디바이스에서 발생하는 데이터 처리를 위한 컴퓨팅, 스토리지 자원 제공 및 관리 중심의 기술에 중점을 두고 있어, 연산 능력을 보유하게된 대규모 엣지 디바이스를 활용하여 다양한 주변 장치를 실시간으로 관리하는 연구는 부족한 상황이다. 최근 분산된 노드를 관리하는 프레임워크로 널리 사용되는 Kubernetes와 같은 컨테이너 기반 클라우드 서버 환경에서도 관리가능한 가상화 단위인 POD를 노드당 110개를 제시하는 등 대부분의 기술이 대규모 엣지 디바이스를 고려하고 있지 않는 상황이다 [2]. Kubernetes는 데이터 센터를 대상으로 하는 기술로써 대부분의 엣지 디바이스에는 완전한 Kubernetes 배포를 지원하기에 충분한 하드웨어 리소스가 없는 경우가 많고 엣지 디바이스와 같이 여러 지역에 대규모로 분산하기 시작하면 네트워킹에 어려움을 겪게 된다. 따라서 본 절에서는 대규모·실시간 동적으로 신규 데이터가 생성되는 환경에서 수많은 엣지 디바이스를 동적으로 관리하는 연구를 진행하기 위한 관련 연구 및 제안하는 연구가 활용될 수 있는 분야의 기술 동향에 대해 살펴보고자 한다.

### 1. 엣지 디바이스 모니터링 기술 동향

다양한 기능을 수행하는 IoT 장치의 확산으로 인해 관련 분야의 다양한 연구가 진행되고 있다. 특히 IoT 장치의 데이터를 수집, 전송, 관리에 중점을 두고 있는 연구와 수집되는 데이터명세 및 활용에 집중하는 연구가 주로 진행되고 있다. 데이터 수집 및 관리 분야에서는 IoT 장치의 영상 정보를 수집하고 모니터링 작업을 수행하는 시스템을 구축하기 위해 경량화된 Kubernetes인 K3S를 활용하여 라즈베리파이와 같은 저사양의 시스템에 서버 클러스터를 구성하고 RabbitMQ와 InfluxDB를 통해 관리하는 연구가 있다 [3]. 또한 엣지와 클라우드 플랫폼이 애플리케이션 호스팅의 기반이 되어감에 따라 Kubernetes 클러스터를 구축하고 이를 통해 애플리케이션과 리소스를 모니터링할 수 있는 Prometheus기반 모니터링에 관한 연구도 있다 [4]. 가정 환경에서 건강 문제의 원격 모니터링 및 조기 탐지를 위해 IoT 장치를 통합하여 혈중 산소 수준, 심박수, 체온 및 ECG 신호를 포함한 중요한 생리학적 데이터를 수집하고 MQTT 프로토콜을 사용하여 서버로 전송하여 분석하는 연구가 있다 [5]. 수집되는 데이터의 정형화 분야에서는 다양한 IoT 디바이스가 존재하는 현실에서 빠르고 효율적으로 이들을 각각 관리하기 위해 디바이스의 등록과 관리를 위한 디바이스 레지스트리를 활용하는 모니터링 연구가 있으며, IoT 및 클라우드 환경에서 SLO (Service Level Objectives)와의 일치를 보장하기 위해 주기적인 자원 평가를 가능하게 하는 SLA (Service Level Agreement)지원을 위한 포괄적인 데이터 수집 연구도 있다 [6]. 그러나 대부분의 연구가 초기 설정된 IoT 장치의 데이터 수집 및 전송에만 집중되어 있어 분산된 대규모 장비들을 관리하는 것이 복잡해질 수 있으며,

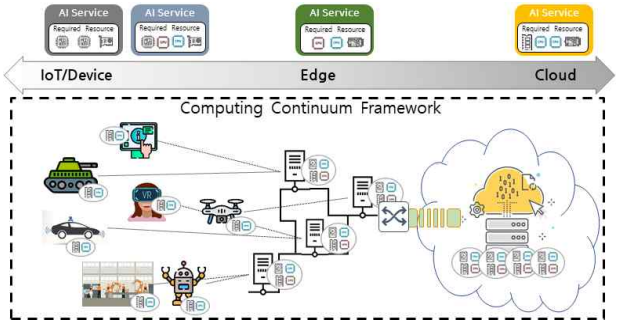



그림 1. 컴퓨팅 컨티뉴엄 개념도
Fig. 1. The Concept of Computing Continuum

시스템 간의 통합 문제 발생 가능성이 있다. 또한 Kubernetes, OpenStack과 같은 기존 환경을 그대로 활용한 클러스터 구조 구성으로 추가적인 디바이스나 기능 확장이 용이하지 않고 자원을 효율적으로 사용하는데 한계가 있다. 본 논문에서는 대규모 디바이스 관리를 위한 디바이스 메타 모델을 정의하고, 클라우드 자원 상태와 상호작용하기 위한 지능형 에이전트를 설계하여 SLO에 기반하여 정의된 메트릭과 정책에 따라 대규모 엣지 디바이스를 안정적으로 관리할 수 있는 연구를 진행한다.

### 2. IoT-엣지-클라우드 연계 컴퓨팅 기술

IoT-엣지-클라우드 연계 컴퓨팅 기술은 임베디드 장치와 IoT의 발전에 힘입어 새롭게 등장한 기술이다. 저사양의 IoT에서 인공지능과 같은 고성능의 서비스를 동작시키기 위해 엣지 및 클라우드와의 자동적인 연계를 지원한다. 이를 위해서는 IoT-엣지-클라우드로 구성된 환경에서 각각의 노드에서 생성·관리되는 데이터를 어디서든 자유롭게 활용할 수 있어야 하며, 지속적으로 관리되어야 한다. 최근 IoT-엣지-클라우드 연계 컴퓨팅 기술은 컴퓨팅 컨티뉴엄 (Computing Continuum)이라는 용어로 정의되고 있으며, 컴퓨팅 기술의 발전에 따라 다양한 디바이스와 이질적인 운영환경을 연속적으로 통합하고 발전시키는 개념으로 확산되고 있다 [7, 8].

그림 1은 IoT/Device에서부터 엣지와 클라우드까지 연계하여 서비스를 제공하는 컨티뉴엄의 개념을 보여준다. 이를 위해서는 전통적인 데스크탑 컴퓨터부터 스마트폰, 웨어러블 디바이스, IoT 장치, 클라우드 서비스, 엣지 컴퓨팅에 이르기까지 다양한 컴퓨팅 장치들 간의 상호작용과 데이터 공유가 원활하게 이루어져야 한다 [9, 10]. 따라서 대규모 엣지 디바이스의 데이터를 안정적으로 수집 및 관리하는 본 연구는 컴퓨팅 컨티뉴엄을 구현하기 위해 필수적으로 연구되어야 한다.

### 3. 디바이스리스 컴퓨팅 기술

디바이스리스 컴퓨팅 (Deviceless Computing)은 디바이스 운영·관리 및 용적 계획 결정이 개발자나 운영자로부터 완전히 숨겨지는 개념으로, 개발자 중심의 혁신 서비스 개발,

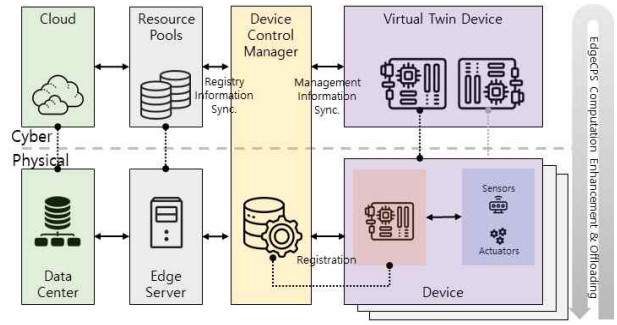



그림 2. 디바이스리스 컴퓨팅 구조
Fig. 2. Deviceless Computing Architecture

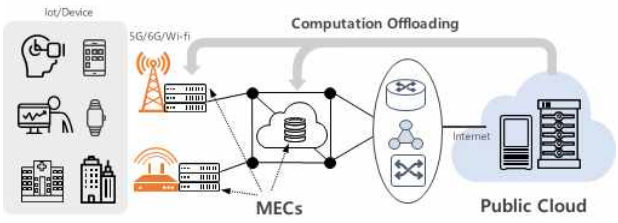



그림 3. MEC의 개념
Fig. 3. The Concept of MEC

비용 효율적인 운영, 탄력적인 확장성, 안정적인 운영 환경 등의 장점을 제공한다 [11]. 이는 서버리스 컴퓨팅과 대비되는 개념으로, 기존의 중앙집중식 클라우드 컴퓨팅 모델인 서버리스 컴퓨팅과는 달리 데이터 처리와 컴퓨팅 작업을 네트워크의 가장자리에 위치한 엣지 디바이스를 중심으로 수행하는 방식을 채택한다. 디바이스리스 컴퓨팅은 서버리스 컴퓨팅의 장점을 디바이스까지 확장함으로써, 물리적인 디바이스는 필요하지만 플랫폼에서 모든 번거로운 작업을 처리하게 된다. 이로 인해 서비스가 실행되는 디바이스의 하드웨어나 운영체제에 대해 알 필요가 없어지며, 개발자와 운영자는 개별 디바이스의 성능이나 사양을 고려하지 않고도 일관된 방식으로 개발, 운영 및 관리할 수 있다.

그림 2에서 디바이스리스 컴퓨팅에 대한 개념도를 보여준다. 그림과 같이 사용자 서비스 요청이 발생하면 엣지 디바이스는 SLO에 따라 최적의 실행 방법을 찾아 수행해야 하기 때문에 엣지 디바이스의 데이터를 수집·관리하는 것은 매우 중요한 요소이다 [6]. 따라서 본 논문에서는 디바이스리스 컴퓨팅과 같은 새로운 디바이스 운영 환경에서도 쉽게 활용이 가능한 기술로 연구를 진행했다.

### 4. MEC (Multi-access Edge Computing) 기술

최근 급증하는 디바이스로 인한 폭발적인 데이터 증가와 자율주행차, 스마트팩토리 등 초저지연·초연결 특성이 필요한 융합산업서비스의 등장으로 인해 중앙 집중방식의 클라우드 컴퓨팅만으로는 사용자 요구를 만족시킬 수 없는 상황에 도달했다. 이러한 문제를 해결하기 위해 네트워크의 가장자리에서 클라우드와 같은 역할을 수행하는 MEC기술이 등장하게 되었으며, 무선 네트워크 사업자 및 이동통신 사업자는 자사의 망내에 MEC를 배치하여 빠르고 안정적인 서비스를 제공하는 동시에 새로운 비즈니스 창출의 기회로 보고 있다. 그림 3과 같이 MEC는 컴퓨팅 자원을 최종 사용자 단말과 지리적으로 가까운 위치에 배치하여 서비스를 처리함으로써 보다 안정적이고 빠른 응답시간을 보장하여 사용자 QoE (Quality of Experience)를 향상시킬 수 있는 기술이다 [12]. 특히 MEC는 컴퓨팅 컨티뉴엄 패러다임과 디바이스리스 컴퓨팅을 구현하기 위해 필수적인 요소로 활용될 것이며, 대규모 IoT가 설치된 환경에서 초대규모·초저지연 서비스를 제공할 수 있어 다양한 융합서비스에서 요구하는 AI, 빅데이터, CPS 등을 구현하기 위해 필수적인 기술로 활용될 것이다. 본 논문에서도 제안한 기술을 구현하기 위해 향후 대규모 디바이스가 설치되는 환경에 필수적으로 사용될 MEC를 활용하고자 하여 구현하고자 한다.

### 5. KubeEdge

KubeEdge는 Cloud Native Computing Foundation (CNCF)에 의해 지속적으로 관리 및 개선되는 Kubernetes기반의 경량화된 컨테이너 오케스트레이션 프레임워크이다 [9]. 이 프레임워크의 주요 장점 중 하나는 클라우드 연결이 불안정하거나 끊어지는 상황에서도 엣지 디바이스에서 서비스의 지속적인 운영을 보장할 수 있는 능력에 있다. 이는 분산된 컴퓨팅 환경에서 높은 신뢰성을 제공함을 의미한다. 또한, KubeEdge는 다양한 종류의 엣지 디바이스와의 원활한 통합을 목표로 하며, 이를 위해 장치 친화적인 구조를 제공한다. 이러한 구조는 개발자들이 다양한 하드웨어 환경에 맞춰 서비스를 손쉽게 배포하고 관리할 수 있도록 지원한다. 그러나 KubeEdge는 엣지 디바이스를 kubetnetes의 노드로 활용하기 위한 기술이기 때문에 노드로 직접 활용되기 어려운 다양한 페리퍼럴들의 데이터를 관리하고 효율적으로 운영하는 기술은 부재한 상황이다. 본 논문에서는 KubeEdge를 기반 인프라로 활용하여 페리퍼럴의 데이터를 수집 및 관리할 수 있는 디바이스 관리기를 구현하고자 한다.

## Ⅲ. 대규모 엣지 디바이스 관리기 설계

### 1. 설계 고려사항

IoT 장비의 기하급수적인 증가와 고용량, 고품질의 다양한 서비스 요구로 인해 엣지 디바이스의 연결, 자원 관리, 자원 가상화, 자원 동적 관리 등을 위한 기술이 필요해지고 있다 [13-16]. 그러나 기존에는 클라우드와 센서 노드의 단순 연결에만 중점을 두고 기술 개발이 이루어져 왔기 때문에 대규모 엣지 디바이스 정보 및 디바이스에 연결된 센서·액추에이터 같은 페리퍼럴의 정보를 관리하는 기술이 부재한 상황이다. 또한 엣지 디바이스가 연산능력을 가진 디바이스로 진화됨에 따라 메모리, 가속기 등의 리소스와 도메인 별로 다양한 종류의 페리퍼럴을 장착하게 되었으나 현재는 이러한 상황을 고려하지 못하고 있다. 따라서 본 절에서는 대규모 엣지 디바이스를 활용하는 다양한 환경에서 사용

할 수 있는 엣지 디바이스 관리기를 제안한다. 엣지 디바이스 자원관리를 위한 엣지 디바이스 관리기는 연결되는 장치 및 자원들에 대한 관리가 필수적이나 아래와 같은 요소를 고려하여 설계해야 한다.

1. 자원의 통합 관리를 위해 컴퓨팅 성능을 갖춘 디바이스와 디바이스에 연결되는 페리퍼럴 (Peripheral)을 통합하여 관리한다.
2. CPU, GPU, MEM, Disk 등의 시스템 내부 자원들도 디바이스에 연결되는 물리적 자원과 동일한 형태로 관리할 수 있도록 구성한다.
3. 디바이스별로 통신을 위한 맵퍼 (Mapper)를 제공함으로써, 다양한 통신 프로토콜에 대응하고 외부 자원에 접근할 수 있도록 연결관리기 모듈을 설계한다.
4. 엣지 디바이스는 사용자 개인정보 및 민감정보를 포함하고 있으므로, 누구나 접근 가능한 인터넷에 연결된 클라우드를 사용하는 것은 매우 부적절하다는 점을 고려한다.

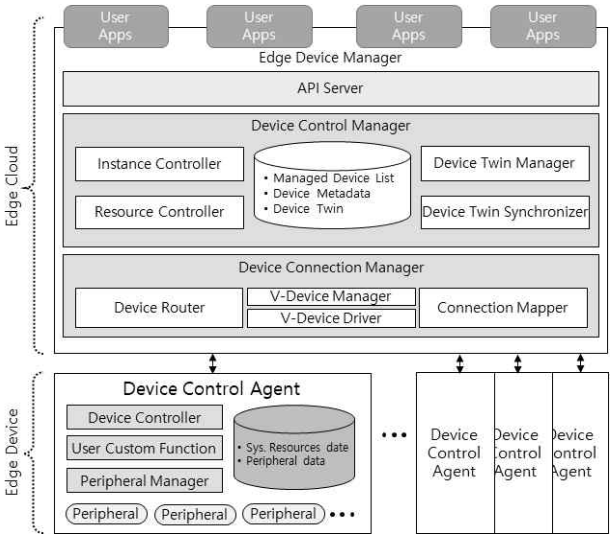



그림 4. 엣지 디바이스 관리기 구조

Fig. 4. Edge Device Manager Architecture

## 2. 시스템 구조

엣지 디바이스 관리기는 연결된 디바이스의 페리퍼럴 정보를 바탕으로 각 페리퍼럴들의 성능 및 현재 상태 등을 종합적으로 모니터링할 수 있어야 한다. 그림 4는 본 논문에서 제안하는 엣지 디바이스 관리기의 구조를 보여준다. 이 프로토타입은 엣지 디바이스 관리기를 포함하며 엣지 디바이스를 위한 모니터링 기능을 효율적으로 구현했다. 엣지 디바이스 관리기는 크게 3가지의 모듈로 구성된다. API server는 사용자 응용 프로그램과 통신을 위한 모듈로 다양한 응용 프로그램과의 원활한 통신을 위해 RESTful API를 제공한다. 디바이스 제어 관리기 (Device Control Manager)는 실제 디바이스와 동일한 가상의 디바이스 트윈을 구성하여 관리하는 모듈로 사용자 응용 프로그램이 실제 디바이스에 대한 물리적인 정보를 알지 못해도 디바이스 트윈을 통해 정형화된 방법으로 디바이스를 활용할 수 있게 해준다.

이를 위해 디바이스 제어 관리자는 관리하는 디바이스에 대한 메타 정보를 저장하고 있으며, 실시간으로 동기화하여 실제 디바이스와 디바이스 트윈간의 동일한 디바이스로 보이게 만든다. 또한, 인스턴트 컨트롤러는 각 페리퍼럴의 속성과 등록을 담당하며 리소스 컨트롤러는 등록된 페리퍼럴의 정보를 통합적으로 관리할 수 있도록 제어하는 기능을 담당한다. 디바이스 연동 관리기 (Device Connection Manager)는 실제 디바이스와 물리적인 연동을 지원하는 모듈로써 다수의 디바이스를 관리하는 기능과 가상의 디바이스 드라이버를 생성하여 실제 디바이스에 탑재되어 있는 페리퍼럴의 정보를 동기화시키는 기능을 제공한다. 연동 맵퍼는 디바이스와의 연동을 위해 다양한 프로토콜을 사용할 수 있게 해준다. 실제 사용되는 페리퍼럴의 데이터를 메시지 토픽에 맞게 변환하기 위한 일종의 함수를 포함하여 Modbus, OPC-UA, Serial, GPIO 등 다양한 형태로 전달되는 데이터를 일률적인 형태로 변환한다. 이는 사용자가 구현하는 응용 서비스에서 동일한 메타데이터 기반 모니터링 및 관리를 위해 필수적으로 필요한 부분이다. 이러한 동작을 위해 실제 엣지 디바이스에는 엣지 디바이스 관리기와 연동을 위한 엣지 디바이스 제어 에이전트 (Device Control Agent)가 존재한다. 엣지 디바이스제어 에이전트는 디바이스에 설치된 페리퍼럴의 관리 및 데이터 동기화를 담당하며 디바이스의 시스템 정보와 페리퍼럴 정보 데이터의 생성, 읽기, 갱신, 삭제 (CRUD: Create, Read, Update, Delete)기능을 제공한다. 또한 사용자별로 엣지 디바이스에 요구하는 기능을 수행하는 기능도 제공한다. 본 논문에서는 대규모 디바이스가 설치된 환경에서 안정적인 엣지 디바이스 관리를 위해 Kubernetes와 KubeEdge 프레임워크를 기반으로 한 엣지 디바이스모니터링 프레임워크의 프로토타입을 설계하였다 [17, 18]. 엣지 디바이스 관리기는 다수의 엣지 디바이스를 통합적으로 연동하고 실제 엣지 디바이스와 매칭되는 디바이스 트윈을 생성 및 관리하기 위해 MEC로 구성된 엣지 클라우드에 Kubernetes를 활용하여 구현하였으며, 실제 엣지 디바이스별로 탑재되는 디바이스 제어 에이전트는 KubeEdge를 활용하여 구현하였다.

표 1은 본 프레임워크를 통해 활용할 수 있는 디바이스 메타정보의 예를 보여준다. 엣지 디바이스 메타정보는 크게 시스템 정보와 디바이스에 설치된 페리퍼럴 정보, 서비스 정보로 구분할 수 있으며, 세부적으로는 표 1과 같이 정의할 수 있으며 사용자 응용 서비스의 요청에 따라 다양한 메타 정보가 포함될 수 있다. 특히 다양한 서비스의 등장으로 인한 사용자 혼란 및 불만족을 극복하기 위해 각 서비스별 품질을 측정하고 관리하기 위한 방법이 필요하며 이를 통해 사용자 만족도가 높은 서비스의 운영에 보다 많은 자원을 투입해야 할 것이다. 표 1의 메타정보 중 SLO는 서비스 제공자가 사용자에게 제공할 서비스의 특정 성능 목표를 정의한 것으로 서비스의 품질을 측정하고 관리하는 데 중요한 역할을 한다. SLO는 SLA와 관련이 있으며 성능을 측정하

표 1. 엣지 디바이스 메타정보의 예
Table 1. Example of Edge Device Metadata

| Category | Description |
|---|---|
| System Information | Device Type |
| | CPU Architecture |
| | Accelerator Type |
| | Memory Capacity |
| | CPU Usage |
| | Memory Usage |
| | Disk Usage |
| | Network Traffic |
| | Temperature |
| | Process List |
| | Power Status |
| | . . . |
| Peripheral Information | Peripheral List |
| | Peripheral ID |
| | Peripheral Type |
| | Peripheral Status |
| | Peripheral Resource Data (size, update cycle, etc.) |
| | . . . |
| Service Information | Service List |
| | Service ID |
| | Service Requirement |
| | Service Tolerance |
| | SLO Information |
| | . . . |

기 위한 응답 시간, 가용성, 처리량 등 다양한 요소를 포함할 수 있다. 또한 특정 기간 내에 달성해야 하는 목표 및 서비스에 대한 성능 목표를 명확히 정의해야 한다 [19]. 즉, SLO를 설정하고 관리함으로써 사용자의 기대를 충족시키고 비즈니스 목표를 달성할 수 있을 것이며, 이를 통해 서비스 수준을 지속적으로 개선할 수 있을 것이다. 결과적으로 표 1에서 보여주는 엣지 디바이스의 모든 메타정보는 엣지 디바이스 관리기에서 디바이스 트윈을 통해 실시간으로 관리 및 동기화된다.

### 3. 동작 절차

엣지 디바이스 관리기를 구현하기 위해 전체적인 동작 절차를 그림 5과 같이 설계했다.

사용자 응용 프로그램에서 모니터링하기를 원하는 대상 디바이스에서 원하는 정보를 획득하기 위해서는 엣지 디바이스 관리기에서 실시간으로 관리하는 정보인지 확인하여 정보를 제공한다. 만일 관리대상이 아닌 경우에는 대상 디바이스와 협력하여 신규 관리정보로 등록한 후 사용자에게 정보를 제공해야 한다. 만일 페리퍼럴 정보인 경우는 일반적인 시스템 정보와 다르게 디바이스 연동 관리기에서 가상 디바이스 드라이버를 생성하고, 메타 정보와 맵핑시켜 일반적인 시스템 정보와 동일하게 처리할 수 있다. 이를 통해 페리퍼럴을 별도의 장치로 인식하게하여 페리퍼럴이 설치된 엣지 디바이스와 관계없이 독립적으로 동작하게 할 수 있고, 다양한 페리퍼럴을 연동가능하게 해준다. 이러한 동작을 통

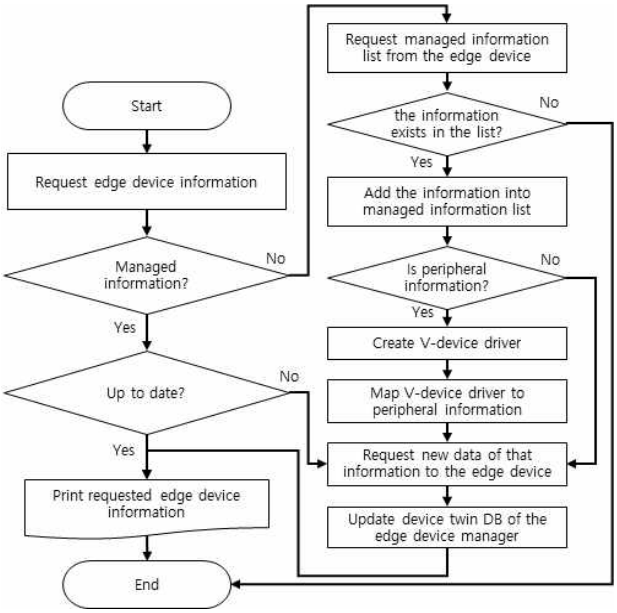



그림 5. 엣지 디바이스 정보 획득 절차
Fig. 5. Device Information Acquisition Procedure

해 향후 엣지 디바이스 관리기에서 관리하는 정보와 실디바이스 정보의 동기화가 시스템 정보 및 페리퍼럴 정보의 구분없이 자동으로 이루어질 수 있으며, 다중 사용자의 요청이 있는 경우 반복적인 데이터 획득 로드를 감소시킬 수 있다.

## Ⅳ. 엣지 디바이스 관리기 구현

### 1. 연동 인터페이스

엣지 디바이스 관리기는 대규모 엣지 디바이스와의 빠르고 안정적인 연동 및 보안·민감 정보의 외부 유출을 방지하기 위해 엣지 디바이스가 설치되어 있는 대상 사이트에 on-site 형태로 구축해야 하며 대규모 디바이스를 관리하기 위한 성능의 문제를 해결해야 한다. 따라서 본 논문에서는 기존의 연구를 통해 개발한 GigaMEC를 활용하여 엣지 디바이스 관리기를 구현하였다. GigaMEC는 ETRI에서 개발한 MEC로써 이동통신사의 5G 네트워크 또는 Wi-fi 네트워크의 말단부인 기지국 및 AP에 배치되어 초저지연 서비스를 제공할 수 있는 플랫폼이다 [7, 20]. GigaMEC는 저지연·고신뢰 서비스를 지원하는 ETSI (European Telecommunications Standards Institute) 표준기반 MEC 플랫폼과 응용 서비스 개발을 지원하는 SDK로 구성된다. GigaMEC는 클라우드 환경을 특정 환경에 구축하여 데이터보안, 빠른 응답시간을 제공하는 기술로 엣지 디바이스를 위한 별도의 기능을 가지고 있지 않으며, 대규모 디바이스에 대한 관리도 지원하지 않는다. 본 연구에서는 Kubernetes를 기반으로 개발된 GigaMEC와 연동하기 위해 엣지 디바이스에는 KubeEdge를 사용하고, 엣지 디바이스 관리기를 통해 엣지 디바이스에 있는 다양한 형태의 데이터를 모니터링하기 위해서 KubeEdge에서 제공하는 MQTT Message Topic을 활용했다.

표 2. KubeEdge 메시지 토픽 구성
Table 2. KubeEdge Message Topic

| Prefix | Type | Content |
|---|---|---|
| $hw/events | node/+ | Device Join Check |
| | device/+/update | Device Status Check |
| | device/+/twin | Data Sync |

표 3. 엣지 디바이스 관리기를 위한 RESTful API
Table 3. RESTful API for Edge Device Manager

| Resource | RESTful API |
|---|---|
| /nodes | Get : node Info. Read<br>Post : node Info. Creation<br>Delete : node Info. Deletion |
| /nodes/:nid | |
| /nodes/:nid/resources | Get : res Info. Read<br>Post : res Info. Creation<br>Put : res Info. Update<br>Delete : res Info. Deletion |
| /nodes/:nid/resources/:rid | |

표 2는 KubeEdge에서 제공하는 기본 Message Topic 중 실제 디바이스가 활용하는 토픽을 보여준다. 각 토픽의 접두사 (Prefix)를 통해 디바이스에 관한 메시지임을 확인하고 종류 (Type)를 통해 각 엣지 디바이스의 접속 및 상태를 확인한 후 데이터를 업로드, 다운로드하여 실제 엣지 디바이스의 정보를 관리할 수 있다. 또한 CPU, GPU, Memory 등의 시스템 정보도 추상화된 디바이스 형태로 변환하여 동일한 구성을 통해 모니터링할 수 있다. 엣지 디바이스의 시스템 데이터 외에 가장 많은 데이터가 생성되는 페리퍼럴 데이터는 메타데이터와 매핑되는 메시지 토픽으로 구성하며, 해당 토픽 정보를 통해 실시간으로 확인할 수 있다. 이를 위해 디바이스 제어 에이전트는 엣지 디바이스 관리자와 연동하여 메타데이터의 관리를 위한 생성, 읽기, 갱신, 삭제 (CRUD) 기능을 제공한다. 표 3은 본 프레임워크에 연결된 엣지 디바이스 정보와 페리퍼럴 정보를 손쉽게 모니터링 및 관리하기 위해 사용자가 외부에서 사용할 수 있는 Restful API 구성을 보여준다. 컴퓨팅 능력이 있는 엣지 디바이스는 실제 서비스가 동작가능한 하드웨어 노드로 표현하며, 컴퓨팅 능력이 없는 페리퍼럴을 노드의 자원과 동일한 형태로 관리하여 제공한다.

## 2. 구현 및 검증

본 논문에서는 엣지 디바이스의 CPU, GPU, MEM 등의 시스템 메트릭 정보와 엣지 디바이스에 연결된 페리퍼럴 정보는 동일한 형태의 추상화된 메타데이터로 확인할 수 있도록 Kubernetes와 KubeEdge의 구조적 특징을 활용하여 CRD (Custom Resource Definition) 형태로 정의했다. 그림 6은 전체적인 구조를 보여준다. 엣지 디바이스에 설치되는 디바이스 제어 에이전트는 KubeEdge의 EdgeCore와 통신하며 동작하고, 관리되는 모든 데이터는 KubeEdge의 Local DB를 활용하여 처리한다. 엣지 디바이스 관리는 MEC에 설치되면 Kubernetes의 K8S API Server를 통해 CloudCore와

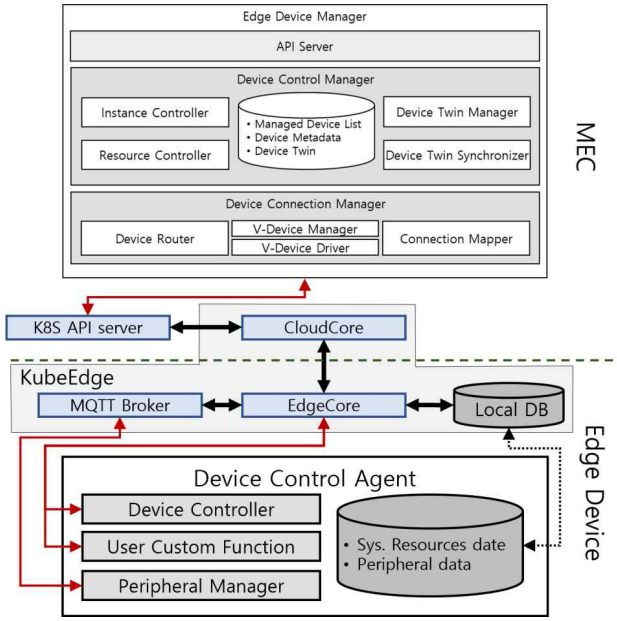
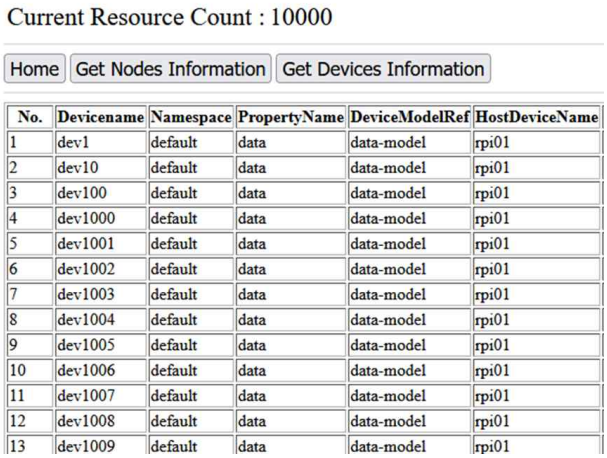



그림 6. 구현 환경
Fig. 6. Implementation Environment




| No. | Devicename | Namespace | PropertyName | DeviceModelRef | HostDeviceName |
|---|---|---|---|---|---|
| 1 | dev1 | default | data | data-model | rpi01 |
| 2 | dev10 | default | data | data-model | rpi01 |
| 3 | dev100 | default | data | data-model | rpi01 |
| 4 | dev1000 | default | data | data-model | rpi01 |
| 5 | dev1001 | default | data | data-model | rpi01 |
| 6 | dev1002 | default | data | data-model | rpi01 |
| 7 | dev1003 | default | data | data-model | rpi01 |
| 8 | dev1004 | default | data | data-model | rpi01 |
| 9 | dev1005 | default | data | data-model | rpi01 |
| 10 | dev1006 | default | data | data-model | rpi01 |
| 11 | dev1007 | default | data | data-model | rpi01 |
| 12 | dev1008 | default | data | data-model | rpi01 |
| 13 | dev1009 | default | data | data-model | rpi01 |

그림 7. 엣지 디바이스 연동 시험 결과
Fig. 7. Edge Device Connection Test Result

연동된다. 또한 외부 사용자 응용 서비스의 접근성을 향상시키기 위해 RESTful POST, GET, PUT, DELETE API를 구축하여 대규모 엣지 디바이스에 적용할 수 있게 개발했다.

그림 7은 제시한 구조에 따라 구현된 환경에서 golang으로 개발된 kube-client 모듈과 RESTful API를 바탕으로 API server를 통해 10,000개의 연결 요청을 처리한 결과를 보여준다. 각 엣지 디바이스의 정보는 KubeEdge 토픽과 연결된 엣지 디바이스 ID (rid)로 구분되며 엣지 디바이스 전체 혹은 각 엣지 디바이스별로 구분되어 조회할 수 있다. 또한, 엣지 디바이스 관리기와 RESTful API에 각각 구현된 CRUD 함수들을 통해 원격지에서 생성 및 데이터 활용이 가능하고, 10,000개 이상의 엣지 디바이스 정보를 동시에 관리할 수 있었다.

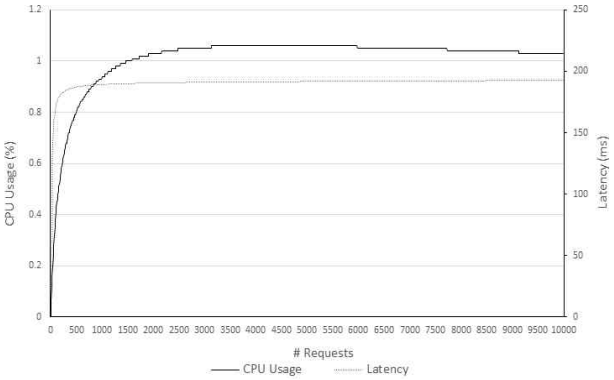



그림 8. 엣지 디바이스 관리기의 성능 측정 결과
Fig. 8. Performance Measurement of Edge Device Manager

그림 8은 엣지 디바이스 관리기의 성능 측정 결과를 보여준다. 이번 실험을 위해 엣지 디바이스 관리기는 CPU는 Intel 12세대 i9-12900KS, 메모리는 64GB, 네트워크는 1000M 이더넷으로 구성된 시스템에 구축했고, 디바이스 제어 에이전트는 페리퍼럴들이 장착된 라즈베리파이 4B/4GB에 설치했다. 모든 페리퍼럴을 실제 시스템으로 제작할 수 이 없기 때문에 실제 시스템과 라즈베리파이에서 추상화된 가상의 디바이스를 혼합하여 10,000의 요청을 발생시켜 엣지 디바이스 관리자의 CPU 사용량과 처리 지연시간(Latency)을 측정하였다. CPU 사용량은 평균 1.01%이며, 처리 지연시간은 10,000개의 반복된 서비스 처리 요청에 의해 지속적으로 증가하고 있으나, 요청횟수에 따라 급격하게 증가하지 않고 평균 190.5ms의 안정된 상태로 동작했다. 이번 실험을 통해 본 논문에서 제시한 엣지 디바이스 관리기는 실험실 환경에서 일반적인 사양의 시스템으로도 안정적인 운영을 할 수 있음을 증명했다. 그러나 실제 운영환경에 적용시에는 도메인별로 다양한 성능 요구사항과 시스템 구성이 상이하고 다양한 페리퍼럴이 존재하기 때문에 약간의 성능저하가 발생할 수도 있을 것이다. 따라서, 향후 연구에서는 특정 도메인을 선정하여 실적용을 추진하고 이에 따라 발생하는 문제점을 해결하여 활용 가능성을 높이는 연구를 진행하고자 한다.

## V. 결 론

컴퓨팅 기술의 발전과 5G와 같은 초고속 이동통신망의 등장은 현실세계에서 수많은 엣지 디바이스의 활용도를 극적으로 높여주고 있으며, 미래에는 응용 분야가 더욱 다양해지고 그 수요도 지속적으로 증가할 것이다. 이에 따라 기존에 구축되어 있는 엣지 디바이스의 기능과 성능을 실시간으로 모니터링하고 그 결과를 바탕으로 엣지 디바이스의 제어 및 복잡한 서비스 배포를 가능하게 하는 기술의 필요성이 급증하고 있다. 본 논문에서는 기존 기술의 한계로 폭넓게 활용되지 못했던 단순한 센서 및 저사양·저자원 디바이스를 MEC와 연계하여 활용하기 위한 대규모 엣지 디바이스 모니터링 기술에 대해 제안했다. 특히 대규모 엣지 디바이스를 안정적으로 관리하기 위해 엣지 디바이스 관리기와 디바이스 제어 에이전트를 설계하고 구현하였으며, 10,000개 이상의 엣지 디바이스 정보를 동시에 모니터링하는 결과를 통해 제안한 방법의 활용가능성을 검증하였다.

향후에는 본 논문에서 제안한 모니터링 기법을 바탕으로 수많은 엣지 디바이스가 설치되어 있는 시스템의 전체 효율성 향상 및 도메인별 SLO를 높이기 위한 지능형 스케줄링 및 오케스트레이션 기법에 대한 연구를 진행할 예정이며, 이를 통해 무수히 많은 사물로 구성된 스마트 공간 (공장, 쉽야드, 도심, 차량 등)에 엣지 디바이스의 폭넓은 활용이 가능해져 다양한 신산업 융합서비스 개발에 이바지할 것으로 기대한다.

## References

[1] I. G. Chun, S. J. Kang, G. J. Na, "Trends in EdgeCPS Technology for Autonomous Control of Large-Scale Devices," ETRI Electronics and Telecommunications Trends, Vol. 37, No. 1, pp. 32-41, 2022 (in Korean).

[2] https://kubernetes.io/docs/setup/best-practices/cluster-large/

[3] K. J. Lee, S. Y. Lee, D. W. Jeong, E. J. Lee, S. H. Lee, "IoT Device Monitoring Tool based on Device Registry," Proceedings of KIIT (Korea Institute of Information Technology) Conference, pp. 328-332, 2020 (in Korean).

[4] B. C. Kang, B. S. Kim, "K3s-Based Cluster Implementation for Edge Device Monitoring," Summer Annual Conference of IEIE (The Institute fo Electronics and Information Engineers), 2024 (in Korean).

[5] M. R. Islam, M. M. Kabir, M. F. Mridha, S. Alfarhood, M. Safran, D. Che, "Deep Learning-Based IoT System for Remote Monitoring and Early Detection of Health Issues in Real-Time," Sensors, Vol. 23, No. 11, pp. 5204, 2023

[6] V. K. Prasad, D. Dansana, M. D. Bhavsar, B. Acharya, V. C. Gerogiannis, A. Kanavos, "Efficient Resource Utilization in IoT and Cloud Computing," Information, Vol. 14, No. 11, pp. 619, 2023

[7] S. J. Kang, I. G. Chun, "Computing Continuum: Trends in Embedded-Edge-Cloud Computing Integration Technologies," IITP Weekly ICT Trends, No. 2023, pp. 2-13, 2021 (in Korean).

[8] T. Sipola, J. Alatalo, T. Kokkonen, M. Rantonen, "Artificial Intelligence in the IoT Era: A Review of Edge AI Hardware and Software," 2022 31st Conference of Open Innovations Association (FRUCT), pp. 320-331, 2022

[9] S. Dustdar, V. C. Pujol, P. K. Donta, "On Distributed Computing Continuum Systems," IEEE Transactions on Knowledge and Data Engineering, Vol. 35, No. 4, pp. 4092 - 4105, 2022.

[10] P. K. Donta, I. Murturi, V. Casamayor Pujol, B. Sedlak, S. Dustdar, "Exploring the Potential of Distributed Computing

Continuum Systems," Computers, Vol. 12, No. 10, pp. 198, 2023.
[11] S. Nastic, S. Dustdar. "Towards Deviceless Edge Computing: Challenges, Design Aspects, and Models for Serverless Paradigm at the Edge," The Essence of Software Engineering. SpringerOpen, 2018. pp. 121-136.
[12] J. H. Lee, S. J. Kang, J. H. Jeon, I. H. Chun, "Multiaccess Edge Computing-Based Simulation as a Service for 5G Mobile Applications: A Case Study of Tollgate Selection for Autonomous Vehicles," Wireless Communications and Mobile Computing, Vol. 2020, No. 4, pp. 15, 2020.
[13] https://kubeedge.io/
[14] S. H. Kim, T. H. Kim, "Local Scheduling in KubeEdge-Based Edge Computing Environment," Sensors, Vol. 23, No. 3, pp. 1522, 2023.
[15] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, R. Schober, "Massive Access for 5G and Beyond," in IEEE Journal on Selected Areas in Communications, Vol. 39, No. 3, pp. 615-637, 2020.
[16] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, J. Cao, "Edge Computing with Artificial Intelligence: A Machine Learning Perspective," ACM Computing Surveys, Vol. 55, No. 9, pp. 1-35, 2023.
[17] M. Fogli, T. Kudla, B. Musters, G. Pingen, C. V. den Broek, H. Bastiaansen, N. Suri, S. WebbM, "Performance Evaluation of Kubernetes Distributions (K8s, K3s, KubeEdge) in an Adaptive and Federated Cloud Infrastructure for Disadvantaged Tactical Networks," 2021 International Conference on Military Communication and Information Systems (ICMCIS), pp. 1-7, 2021.
[18] T. Goethals, F. De Turck, B. Volckaert, "Extending Kubernetes Clusters to Low-Resource Edge Devices Using Virtual Kubelets," in IEEE Transactions on Cloud Computing, Vol. 10, No. 4, pp. 2623-2636, 2020.
[19] W. I. Seo, S. H. Cha, Y. J. Kim, J. H. Huh, J. S. Park, "SLO-Aware Inference Scheduler for Heterogeneous Processors in Edge Platforms," ACM Transactions on Architecture and Code Optimization (TACO), Vol. 18, No. 4, pp. 1-26, 2021.
[20] https://gigamec.ai/

**In-geol Chun (전 인 걸)**

1998 Electrical and Computer Engineering from SungKyunKwan University (M.S.)
2011 Electrical and Computer Engineering from SungKyunKwan University (Ph.D.)
1998~ETR (Electronics and Telecommunications Research Institute) (Principal Researcher)

Career:
2012~2021 Professor, ICT from UST (University of Science and Technology)
2018~ Executive Director, IeMeK
2014~ Chairman of CPS Project Group (PG609), TTA

Field of Interests: EdgeCPS, Autonomic Computing system, Hybrid AI, Computing Continuum
Email: igchun@etri.re.kr

**Jong-soo Seok (석 종 수)**

2012 Computer Science and Engineering from Konkuk University (B.S.)
2014 Computer Science and Engineering from Konkuk University (M.S.)
2014~ETRI (Electronics and Telecommunications Research Institute) (Senior Researcher)

Field of Interesst: Operating Systems, Embedded Computing, Cloud Computing
Email: jsseok@etri.re.kr