MEC 기반 긴급 데이터 우선 처리를 위한 논리적 네트워크 슬라이싱 및 동적 자원 재할당 기법 설계

위대훈, 노현민
전북대학교

Wdh0209@jbnu.ac.kr, hmnoh@jbnu.ac.kr

Design of Logical Network Slicing and Dynamic Resource Reallocation for Priority-Based Emergency Data Processing in MEC Systems

Dae Hun Wi, Hyunmin Noh

Jeonbuk National Univ.

요 약

본 논문은 IoT 헬스케어 시스템에서 발생할 수 있는 긴급 데이터 처리 지연 문제를 해결하기 위해, 모바일 엣지 컴퓨팅(MEC)과 네트워크 슬라이싱 개념을 기반으로 한 논리적 데이터 분리 및 동적 자원 재할당 기법을 제안한다. 제안된 시스템은 MEC 서버 내에서 긴급 데이터와 일반 데이터를 논리적으로 분리한 후, 긴급 트래픽이 급증할 경우 일반 트래픽의 처리 속도를 조절하여 긴급 데이터의 우선 처리를 지원한다. 또한, 실시간 큐 상태를 모니터링하여 네트워크 상황에 따라 전송 정책을 유연하게 변경함으로써, 지연 시간을 최소화하고 시스템의 안정성과 자원 활용률을 향상시킨다. 간단한 시뮬레이션 실험을 통해 제안 기법이 실제 환경에서도 동작 가능함을 확인하였다.

Ⅰ. 서 론

IoT 헬스케어 시스템은 환자의 생체 신호를 실시간으로 수집하고 이를 빠르게 분석하여 긴급 상황에 대응하는 기능이 핵심으로 요구되고 있다. 온도, 심박수 등의 중요한 데이터는 지연 없이 전달되어야 하지만, 기존 Wi-Fi 기반 IoT 시스템에서는 모든 데이터가 동일 경로로 전송되어 긴급·비긴급 데이터가 구분 없이 처리되고[1], 트래픽 증가 시 긴급 데이터마저 지연되는 문제가 발생한다. 이를 해결하기 위해 MEC 와 네트워크 슬라이싱 기술이 주목받고 있다. MEC 는 데이터 처리를 네트워크 엣지에서 수행해 지연을 최소화하며, 슬라이싱은 데이터 중요도에 따라 자원을 논리적으로 분리해 긴급 데이터의 우선 처리를 지원한다.

본 연구에서는 긴급 데이터와 비긴급 데이터를 논리적으로 분리하고, 트래픽 증가 시 자원을 동적으로 재할당하는 MEC 기반 시스템을 구현하였다. 실시간 트래픽 상황에 맞춰 전송 정책을 조정하고, 긴급 상황 해소 후 자원을 복구하는 구조를 설계하여 기존 고정 자원 할당 방식의 한계를 보완했다.

Ⅱ. 본론

1) 제안 시스템의 설계 및 구성

본 연구에서는 긴급성과 일반성을 고려한 IoT 헬스케어 데이터의 실시간 처리를 위해, MEC 기반의 논리적 슬라이싱 구조[2]를 구현하였다. IoT 클라이언트, MEC 서버, 클라우드 서버로 구성되며, 각각 데이터 생성, 분류 및 처리, 저장 기능을 분담한다. 그림 1 은 제안 시스템의 전체 구성 흐름을 나타낸다. 그림 1 에서 IoT 클라이언트는 온도와 심박수 데이터를 주기적으로 생성하며, 사전에 정의된 임계값 (예: 온도 ≥ 38℃, 심박수 ≥ 120bpm)에 따라 데이터를 긴급 또는 일반으로 구분한다. MEC 서버는 수신한 데이터를 실시간으로 분석하여 우선순위에 따라 특히 우선 순위 큐 Q_{high} 또는 낮은 순위 큐 Q_{low} 에 분리 저장하고, 처리 순서를 결정한다. 클라우드 서버는 MEC 에서 전처리된 데이터를 수신하여 장기 저장 및 후속 분석의 기반 역할을 한다. 해당 구조는 실제 물리적 네트워크를 분할하지 않고, 논리적 슬라이싱 방식으로 구현되어, 시스템 확장성과 경량성 확보에 유리하며, MEC 에서의 처리 분산으로 긴급 대응 속도를 높일 수 있다. 또한

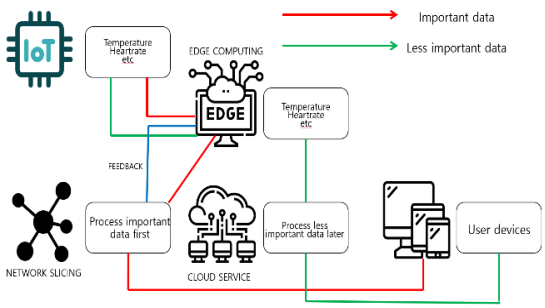

그림 1. 시스템 구조도.

클라이언트를 경량화함으로써 다수의 센서 환경에도 유연하게 대응 가능한 구조를 갖췄다.

2) 실험 시스템 구성 및 구현 방식

본 시스템은 고정된 대역폭 할당이 아닌, 트래픽 상태에 따라 동적으로 자원을 재배분[3]하는 알고리즘을 핵심 처리 방식으로 채택하였다. 특히 우선 순위 큐 Q_{high} 의 트래픽 밀도에 따라 일반 트래픽의 전송 정책을 일시적으로 조정하는 방식으로, 긴급 데이터의 우선 처리를 보장한다.

Step 1) 긴급 트래픽 감지 및 자원 재조정 트리거: Q_{high} 의 길이가 임계값(예: 3 개)을 초과하면, 일반 데이터의 전송 간격(기본 2 초)을 5 초로 조정하여, 시스템 자원을 긴급 데이터에 집중 배치한다.

Step 2) 일정 시간 ($T_{recovery}$ = 5 초) 경과 시 일반 전송 주기를 원래대로 복귀시켜 전체 흐름을 안정화한다..

Step 3) MEC 서버는 항상 Q_{high} 를 우선적으로 비우며, 해당 큐가 비어 있을 경우에만 일반 큐에서 데이터를 처리한다

Algorithm 1: Dynamic Resource Allocation Based on Queue Monitoring

Input: Q_{high}: High Priority Queue,
Q_{low}: Low Priority Queue,
θ: Queue length threshold,
δt_{low}: Low Priority transmission interval (initial 2s),
$T_{recovery}$: Recovery time threshold (5s)
Output: Adaptive data transmission scheduling

$\delta t_{low} \leftarrow 2s$;
$t_{trigger} \leftarrow 0$;
while *True* **do**
  if $|Q_{high}| \geq \theta$ **then**
    $\delta t_{low} \leftarrow 5s$;
    $t_{trigger} \leftarrow$ current time;
  if *current time* $- t_{trigger} \geq T_{recovery}$ **then**
    $\delta t_{low} \leftarrow 2s$;
  while Q_{high} *is not empty* **do**
    Process data from Q_{high} with priority;
  if Q_{low} *is not empty* **then**
    Process data from Q_{low} every δt_{low} seconds;

위 알고리즘은 긴급 데이터와 일반 데이터를 논리적으로 구분하여 처리하며, 큐 상태를 실시간으로 감시하여 트래픽 변화에 따라 자원의 분배를 유연하게 조정한다. 우선, 일반 데이터의 전송 간격 δt_{low} 을 2 초로 설정하고, 긴급 상황 발생 시점을 저장하는 트리거 $t_{trigger}$ 초기화한다. 이후, 시스템은 지속적으로 모니터링한다. 긴급 데이터 큐(의 길이가 사전에 설정된 임계값(Θ) 이상으로 포화되었을 경우, 일반 데이터 전송 간격(δt_{low})을 기존 2 초에서 5 초로 지연시킨다. 이는 제한된 자원을 긴급 데이터 처리에 집중하기 위함이다. 동시에, 긴급 상황이 발생한 시점을 $t_{trigger}$ 변수에 기록한다. 긴급 상황이 일정 시간이상 지속되지 않는 경우, 즉 current time − $t_{trigger}$ ≥ $T_{recovery}$ 조건이 만족되면, 일반 데이터 전송 간격을 다시 원래의 2 초로 복구하여 자원 배분을 정상화한다. 데이터 처리 과정에서는 항상 Q_{high} 를 우선적으로 비우는 작업을 수행한다. 우선 순위 큐가 비어 있지 않은 동안에는 높은 우선순위를 부여하여 지속적으로 데이터를 처리한다. 이후, Q_{low} 에 남아있는 데이터는 설정된 전송 간격 δt_{low} 에 따라 순차적으로 처리된다. 위 알고리즘은 VMware 기반 가상 환경에서 구현되었으며, Python Flask 서버와 시뮬레이션 클라이언트를 통해 데이터를 전송·수신하는 구조로 설계되었다. 이러한 알고리즘은 실시간 큐 모니터링을 기반으로 하며, 전송 간격 조절을 통한 간접적 자원 제어 방식을 구현한다. 실험 결과, 총 120 회의 데이터 전송 과정에서 긴급 트래픽 밀도 증가 상황에서도 긴급 데이터는 실시간으로 안정적으로 처리되었으며, 일반 데이터 전송 간격을 조절하는 방식으로 자원을 효과적으로 재배분할 수 있음을 확인하였다. 전체 평균 지연 시간은 155ms 내외로 측정되었지만, 이는 일반 데이터 전송을 의도적으로 지연시키는 전송 정책의 결과로, 긴급 데이터의 빠른 대응을 위한 우선순위 처리 효과가 반영된 수치이다. 전체 평균 지연시간이 높다고 보일 수 있지만 이는 긴급 처리를 위한 우선순위 전략을 잘 보여준다. 실제로 일시적인 지연 발생 후에도 시스템은 빠르게 원상태로 복귀하여 전체적인 안정성과 자원 활용 효율을 동시에 확보하였다.

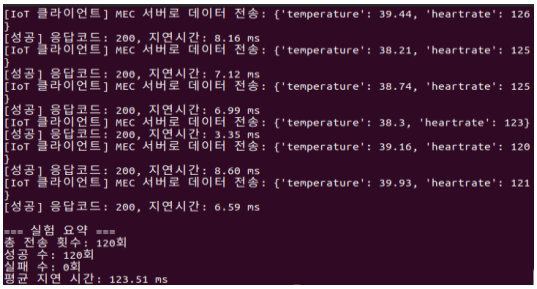

그림 2.결론 측정값

Ⅲ. 결론

본 연구는 기존 IoT 헬스케어 시스템의 네트워크 슬라이싱 기반 구조를 확장해, 긴급 데이터 처리 효율을 높이는 동적 자원 재할당 기법을 구현 및 검증하였다. 기존 논리적 슬라이싱의 고정 자원 한계를 극복하기 위해 MEC 서버 내 실시간 큐 모니터링을 적용, 긴급 트래픽 발생 시 일반 데이터 전송 주기를 조절하는 방식으로 자원을 유연하게 재배분하였다. VMware 가상 환경과 Python 시뮬레이션을 통해 실험한 결과, 긴급상황에서도 안정적 자원 제어와 빠른 복구가 가능함을 확인했으며, 논리적 슬라이싱만으로도 실시간 데이터 처리 성능 향상을 입증하였다. 이는 기존 개념을 실제 동적 트래픽 대응 환경으로 확장한 사례로, MEC 기반 IoT 헬스케어 시스템의 실용성을 높였다는 점에서 의미가 있다. 향후 연구는 자율 슬라이싱 정책을 통한 자원 최적화로 확장할 계획이다.

ACKNOWLEDGMENT .

이 논문은 2025 년도 과학기술정보통신부의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2021-0-00484, 노드 간 메시지 전달과 합의를 위한 최적 경로 네트워크 프로토콜 기술개발)

참 고 문 헌

[1] H. Al-Turjman and M. Malekloo, "An Analytical Model to Minimize the Latency in Healthcare Internet-of-Things," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 4929-4945, 2019.

[2] E. Bërdufi, N. Slamnik-Kriještorac, and J. Marquez-Barja, "Leveraging on Network Slicing to Enable and Enhance IoT-based e-Health Services," *Proceedings of the 2022 Conference on Information Technology for Social Good (GoodIT' 22)*, ACM, pp. 206-211, 2022.

[3] CheifC. Chien, K. Chen, and H. Lin, "Dynamic Resource Allocation for Network Slicing with Multi-Tenants in 5G Networks," Sensors, vol. 23, no. 10, pp. 4698, 2023.