|  |
| --- |
| **MQTT을 활용한 모바일 애플리케이션 엣지환경 구축**  **2016110307 권동영**  **요 약**  모바일 엣지 컴퓨팅은 모바일 클라우드 컴퓨팅이 중앙화된 클라우드 서버를 이용하면서 발생했던 high latency 문제를 사용자와 지리적으로 가까운 엣지 서버를 이용함으로써 해결한다. 엣지 서버는 클라우드 서버와는 달리 여러 장소에 분산되어 있기 때문에 모바일 기기는 자신으로부터 가장 가까이 있는 엣지 서버를 통해 low latency로 컴퓨팅 파워를 제공받을 수 있다. 본 문서는 iOS 애플리케이션과 Raspberry Pi 엣지 서버 간의 MQTT 프로토콜 통신을 하여 모바일 엣지컴퓨팅 환경을 제안 및 구현한다. |

**1. 서론**

**1.1. 연구배경**

현재 IT 분야의 동향을 살펴보면 스마트폰과 태블릿 컴퓨터같은 모바일 기기들의 대부분의 서비스는 클라우드 컴퓨팅 인프라를 활용하여 운영되고 있다. 이에 따라 사림들이 모바일 기기에서 기대하는 성능도 같이 증가하고 있다. 하지만 이러한 성능 향상에도 불구하고5G시대에는 기기가 더 긴밀하게, 방대한 데이터를 주고받으면서, 중앙의 데이터센터에서 모든 데이터를 처리하기에 역부족인 상황이 우려된다. 특히 최근 각광받기 시작한 인공지능(Artificial Intelligence), 증강 현실(Augumented Reality), 클라우드 게이밍(Cloud Gaming) 같은 극단적으로 낮은 전송 지연과 많은 데이터를 요구하는 응용 프로그램을 모바일 기기에서 수행하기는 쉽지않다.

이러한 문제점을 해결하기 위해 모바일 엣지 컴퓨팅(Mobile Edge Computing)이라는 개념이 등장했다. 모바일 엣지 컴퓨팅은 모바일 클라우드 컴퓨팅이 중앙화된 클라우드 서버를 이용하면서 발생했던high latency 문제를 사용자와 지리적으로 가까운 엣지 서버를 이용함으로써 해결한다. 엣지 서버는 클라우드 서버와는 달리 여러 장소에 분산되어 있기 때문에 모바일 기기는 자신으로부터 가장 가까이 있는 엣지 서버를 통해 low latency로 컴퓨팅 파워를 제공받을 수 있다 [1]. 즉, 클라우드 컴퓨팅의 로드를 개별 로컬 서버로 전환함으로써 모바일 네트워크의 혼잡을 줄이고 대기 시간을 줄임으로써 최종 사용자의 QoE (Quality of Experience)를 향상시킨다. MQTT 프로토콜 기술을 통해 모바일 엣지환경을 구축하여 최종 사용자에게 더 가까운 관련 처리 작업을 수행하여 네트워크 정체를 줄인다.

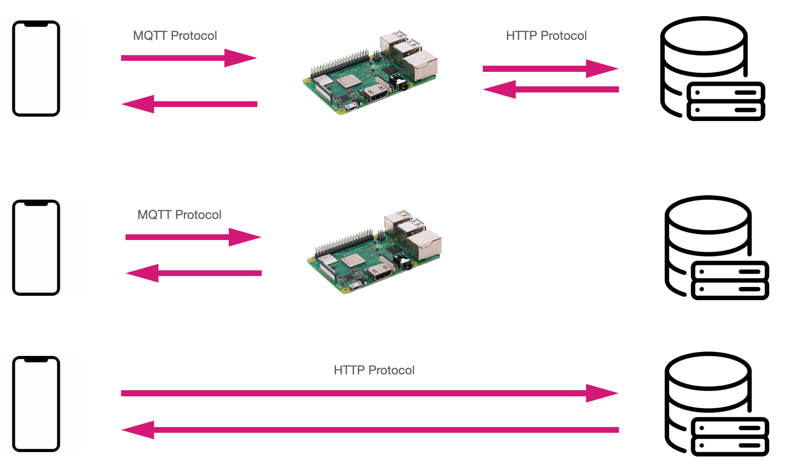
**1.2. 연구목표**

모바일 엣지 컴퓨팅의 핵심은 처리 할 데이터를 원 서버로 오프로딩 하기 전에 연산을 위한 데이터가 미리 엣지 서버에 저장되어 있어야 한다. 이를 위해서 Raspberry Pi를 엣지 서버로 구축하는 것이 첫번째 목표이다.

두 번째 모바일 기기와 엣지서버간에 MQTT 프로토콜 통신을 위해 iOS 응용 프로그램을 개발한다.

세 번째 클라우드 컴퓨팅 인프라를 활용하여 원 서버를 구축한다. 엣지 서버에 연산 데이터가 없는 경우 원 서버로 오프로딩시켜 데이터를 처리한다.

마지막으로 [그림 1]과 같이 1. 엣지서버에 연산 데이터가 없는 경우, 2. 엣지서버에 연산 데이터가 있는 경우, 3. 엣지 서버가 없어 원 서버랑 통신하는 경우, 이 세 가지 경우를 데이터 Latency 비교하여 실험을 진행한다.



**그림 1.**

**2. 관련 연구**

**2.1. MQTT Protocol**

MQTT는1999년 IBM에서 M2M과 사물인터넷에서 사용하기 위해 만들어진 경량의 Publish- Subscribe기반 메시지 프로토콜이다. 사물인터넷의 한계인 낮은 전력, 낮은 대역폭, 낮은 성능 등의 환경에서도 사용 할 수 있도록 설계됐다. 가전기기, 빌딩, 집, 산업 등 다양한 영역에서의 정보를 수집할 수 있다. MQTT는 메시지를 Publish 하고, 관심 있는 주제를 Subscribe 하는 것을 기본 원칙으로 한다. Publish-Subscribe 모델은 센서들 가운데 Broker가 필요하다. Broker는 관심있는 Topic을 기반으로 클라이언트들에게 메시지를 나눠주는 역할을 한다. 이 때 클라이언트는 Publisher와 Subscriber 모두를 지칭한다. Publish와 Subscribe는 Topic을 기반으로 작동한다. Topic은 ‘/’를 이용해서 계층적으로 구성할 수 있어 대량의 센서 기기들을 중복 현상 없이 효율적으로 관리 할 수 있다.

MQTT로 통신을 하기 위해서는 우선 Publisher가 브로커에 연결을 요청해야한다. 연결이 완료 된 후 Publisher는 관심 있는 주제로 브로커에게 정보를 전송 할 수 있다. 만약 연결을 끝내고 싶을 때는 Publisher가 브로커에게 연결 해제 요청을 전송하면 된다. MQTT는 신뢰성을 위해 3가지의 QoS(Quality of Service) 레벨을 정의한다. Qos 0 레벨은 메시지를 최대 1번만 전송 하는 방식으로 메시지가 잘 도착 했는지 중간에 손실 되었는지 알 수 없다. QoS 1 레벨은 메시지를 적어도 한 번 이상 보내는 방식으로 메시지가 잘 도착하면 수신 측에서 ACK 메시지를 전송 한다. 이 ACK 메시지를 보고 Publish/Subscribe가 잘 되었는지 확인 할 수 있다. QoS 2 레벨은 메시지를 정확히 1번 전송 한다. Publish를 하게 되면 수신 측에서 잘 받았으면 PUBACK를 송신 측으로 보내게 되고, 송신 측이 PUBACK까지 잘 받았다면 다시 PUBREL 메시지를 보낸 후 수신 측으로부터 PUBCOMP를 받게 되면 메시지 전송이 완료 된다. QoS 2는 가장 신뢰성이 높은 방식이지만 큰 오버헤드 때문에 낮은 성능

의 기기나 좋지 못한 네트워크 환경에서는 사용하기 힘들 수 있다.



**그림2. MQTT 시스템 구조**

**3. 프로젝트 요구사항**

**3.1 엣지 서버에 대한 요구사항**

모바일 기기에서 발생되는 데이터를 처리하기 위해 엣지 서버를 구축하여야 한다. 손쉽게 구할 수 있는 Raspberry Pi를 통해 엣지 서버 구축을 목표로 한다.

**3.2 모바일 애플리케이션에 대한 요구사항**

엣지 서버와의 MQTT Protocol 통신을 하기 위해 모바일 애플리케이션을 개발하여야 한다. Apple 사의 iPhone 위에서 작동하는 iOS 애플리케이션 개발을 목표로 한다.

**4. 향후 일정**

|  |  |
| --- | --- |
| **날짜** | **내용** |
| 9/13 ~ 9/17 | **지도교수님 면담 및 주제 구체화** |
| 9/18 ~ 9/30 | **주제에 대한 자료조사, 기초조사서 작성** |
| 10/01 ~ 10/08 | **프로그래밍 툴 학습** |
| 10/9 ~ 10/17 | **Raspberry Edge Server, Origin Server 구축** |
| 10/18 ~ 10/25 | **iOS 모바일 어플리케이션 개발** |
| 10/26 ~ 11/02 | **중간보고서 작성** |
| 11/03 ~ 11/12 | **엣지환경속에서 필요한 서비스 조사** |
| 11/13 ~ 11/31 | **서비스 설계 및 구현** |
| 12/01 ~ 12/05 | **기능 테스트 및 오류 수정** |
| 12/06 ~ 12/21 | **최종 보고서 작성 및 최종 발표 준비** |

**5. 결론 및 기대효과**

다가오는 5G 시대에서 기기가 더 긴밀하게, 방대한 데이터를 주고받으면서, 중앙의 데이터센터에서 모든 데이터를 처리해야 한다는 문제점을 해결하기 위해 모바일 엣지 컴퓨팅이 등장하였다. 모바일 엣지 컴퓨팅 환경에서 최종 사용자들이 얼마나 속도향상을 겪은 지를 확인하여 모바일 엣지 컴퓨팅 효과를 기대한다.

**6. 참고 문헌**

[1] 이현재, “모바일 엣지 컴퓨팅 환경에서의 연산 오프로딩을 위한 이동경로 예측”, 공학석사학위논문