ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

대규모 IoT 컴퓨팅 환경에서 동적 클러스터링 기반 효율적 관리 기법☆

A Method for Dynamic Clustering-based Efficient Management in Large-Scale IoT Environment

김 대 영[1] 라 현 정[2*]
Dae Young Kim Hyun Jung La

요 약

IoT 디바이스는 사용자의 주변에 설치되어 네트워크를 통해 다양한 정보를 수집하고, 디바이스간 협업으로 교통 정보를 제공하거나 날씨 정보를 제공하는 등 삶의 질을 향상시키는데 목적이 있다. 현재 여러 연구소나 기업체에서 다양한 IoT 디바이스들이 개발 중에 있어 디바이스의 수가 급격하게 증가하고 있다. 그런데, 이러한 경향에도 불구하고 IoT 디바이스들을 관리하고 연결하는 IoT 컴퓨팅과 관련된 연구는 아직 초기 단계에 있다. IoT 컴퓨팅 환경 내에 대량의 IoT 디바이스가 동시에 존재하게 되는데, 이때 이들 디바이스들을 모니터링 하거나 제어를 위한 관리 노드로서 서버가 필요하다. 그렇지만, 어떤 관리 서버를 어디에 몇 대를 배치해야 하는지 알기 어렵다. 이러한 문제를 해결하기 위해서, 본 논문에서는 대량의 디바이스들을 논리적으로 근접한 디바이스들은 하나의 클러스터로 관리하는 기법을 제안한다. 제안된 동적 클러스터링 기법을 IoT 컴퓨팅 환경에 적용하게 되면 대량의 IoT 디바이스들을 최적의 상태로 클러스터링 함으로써, 디바이스 관리에 대한 오버헤드를 줄이면서 효율적으로 품질 관리를 할 수 있게 된다.

☞ 주제어 : 사물인터넷, 동적 클러스터링, Hierarchical 알고리즘, 최적화, IoT 컴퓨팅 환경

ABSTRACT

IoT devices that collect information for end users and provide various services like giving traffic or weather information to them that are located everywhere aim to raise quality of life. Currently, the number of devices has dramatically increased so that there are many companies and laboratories for developing various IoT devices in the world. However, research about IoT computing such as connecting IoT devices or management is at an early stage. A server node that manages lots of IoT device in IoT computing environment is certainly needed. But, it is difficult to manage lots of devices efficiently. However, anyone cannot surly know about how many servers are needed or where they are located in the environment. In this paper, we suggest a method that is a way to dynamic clustering IoT computing environment by logical distance among devices. With our proposed method, we can ensure to manage the quality in large-scale IoT environment efficiently.

☞ keyword : IoT, Dynamic Clustering, Hierarchical Algorithm, Optimization, IoT Computing Environment

1. 서 론

IoT 디바이스는 옷이나 자동차, 주변 환경 곳곳에 설치되고 네트워크로 서로 연결되어 사용자가 필요로 하는 다양한 정보를 수집한다. 또한, 디바이스간 협업을 통해 교통 정보를 제공하거나 날씨

정보를 제공하는 등 삶의 질을 향상시키는데 그 목적이 있다. 현재 여러 연구소나 기업체에서 다양한 IoT 디바이스들이 개발 중에 있고 디바이스들간 협업을 통해 다양한 서비스도 개발 중에 있다[1][2][3]. 이러한 경향에도 불구하고 IoT 디바이스들을 관리하고 연결하는 IoT 컴퓨팅과 관련된 연구는 아직 초기 단계에 있다.

IoT 컴퓨팅 환경 내에 수십 또는 수만 개의 IoT 디바이스가 동시에 존재하게 되는데, 이때 이들 디바이스들을 모니터링 및 제어를 위한 관리 노드로서 서버가 필요하다. 그렇지만 대량의 디바이스들을 단일 서버를 이용해서 관리한다면 그에 따른 오버헤드 때문에 서버는 효

[1] Department of Computer Science, Soongsil University, Seoul, 156-743, Korea
[2] SmartyLab Corporation, Seoul, 156-725, Korea
* Corresponding author (hjla80@gmail.com)
[Received 12 August 2014, Reviewed 20 August 2014, Accepted 28 August 2014]
☆ 본 연구는 산업통상자원부의 지원을 받는 로봇산업 융합기술개발사업의 연구결과로 수행되었음.

율적으로 디바이스들을 관리하기 어렵다. 그래서 디바이스 관리를 위한 노드를 한대 이상 IoT 컴퓨팅 환경에 배치하여 디바이스를 관리하려고 하는데, 이때 어떤 관리 노드를 어디에 몇 대를 배치해야 하는지 알기 어렵다.

이러한 문제를 해결하기 위해서, 논리적으로 근접한 디바이스들을 하나의 클러스터로 관리하는 기법을 제안한다. 각 클러스터에 하나의 관리 노드가 배치되고, 이들 노드들에 의해서 관리 노드 하나에 집중될 수 있는 관리 오버헤드를 분산시키는 방법을 사용한다. 클러스터링의 결과로 논리적으로 근접한 IoT 디바이스들과 관리 노드를 식별하고 유지할 수 있어 IoT 컴퓨팅 환경에 동적으로 디바이스가 추가되더라도 효율적으로 가장 근접한 클러스터를 찾을 수 있다.

본 논문은, 기반 연구로서 대규모 IoT 컴퓨팅 환경과 동적 클러스터링 기법에 대한 개념을 소개한다. 그리고 IoT 디바이스들을 클러스터에 할당하기 위한 기준인 논리적 근접도에 대해 소개하고, 이를 바탕으로 클러스터링하는 Hierarchical 알고리즘을 IoT 컴퓨팅 환경에 맞도록 일부 수정한 동적 클러스터링 기법을 제안한다. 설계된 기법이 IoT 컴퓨팅 환경에 적용했을 때 클러스터링 결과가 최적으로 진행되었는지 확인하기 위한 평가 모델을 정의하고, 이를 이용하여 각 클러스터의 품질 및 전체 클러스터링 환경을 평가하는 실험을 진행하여 제안된 기법의 효과와 적용성을 확인하였다.

본 논문에서 제안하는 동적 클러스터링 기법을 IoT 컴퓨팅 환경에 적용할 경우 다음과 같은 효과가 기대된다.

- 대규모 IoT컴퓨팅 환경에서의 IoT 디바이스 클러스터링 기초 연구
- IoT디바이스 관리에 대한 오버헤드 감소 기법 제시
- 대규모 IoT컴퓨팅 환경에서의 효율적 IoT디바이스들의 품질 관리 가능

2. 관련 연구

현재 IoT 컴퓨팅 환경을 동적 클러스터링 하여 관리하는 분야의 관련 연구는 선행되고 있지 않다. IoT 분야에서는 IoT 디바이스들을 연결하고 센싱하는 플랫폼 기반의 연구가 대부분 진행 중에 있고, 센서 네트워크 분야에서는 저비용 클러스터링이나 로드 밸런싱과 관련된 연구가 진행되고 있다[4][5][6][7][8][9].

먼저, IoT 관련 프로젝트로서 “Smart Santander” 프로젝트에서는, IoT 기반의 실험적인 연구 도시를 개발하는 것이 목적이다[10]. 스마트한 도시를 만들기 위해 IoT 기반의 여러 애플리케이션과 서비스를 제공한다. 약 20,000여개에 달하는 센서를 통해서 사용자와 주변 환경을 센싱하여 교통 및 날씨 등에 대한 정보를 제공한다. 그렇지만, 이 프로젝트에 활용되는 IoT 디바이스는 센서 위주로서 디바이스의 이동성 또는 컴퓨팅 파워가 있는 유형의 IoT 디바이스는 고려되고 있지 않다. 다른 프로젝트인 “Internet of Things for Creating Smart Cities”는 Melbourne 대학에 의해서 진행 중인 프로젝트이다[11]. IoT 기반의 스마트한 도시를 만들어 소음이나 환경 오염을 모니터링하고, 보안과 개인 정보 보호 등을 세부 목적으로 하고 있다. 그렇지만, 센서 기반의 스마트 도시 구축을 목적으로 하고 있어 다양한 유형의 IoT 디바이스에 대한 관리 연구가 부족하다. Zorzi의 연구에서는 IoT 컴퓨팅 환경에 대한 직면하고 있는 이질성, 보안, 적용 등의 문제들을 소개하고, 특히, IoT 디바이스의 이동성 해결을 위한 방법에 대해 제안했다[12]. 하지만, 대량의 디바이스를 관리해야 하는 상황에 대한 고려가 부족하다.

센서 네트워크 분야의 Gerla와 Parekh의 연구에서는 노드간 근접 정도를 기반으로 클러스터링을 하는 기법을 제시하였다[13]. 한 노드는 전송 가능한 거리에 있는 노드들과 클러스터링되며, 가장 많은 노드의 정보를 가진 노드를 관리자로 지정한다. 하지만, IoT 디바이스와 관련된 요소가 고려되지 않아 IoT 컴퓨팅 환경에 직접적인 적용은 어렵다. Du의 연구에서는 디바이스간 동적 클러스터링 알고리즘을 제안하고 있다[14]. 여러 디바이스들을 클러스터 단위로 분할하고, 새롭게 추가되는 디바이스는 중앙 관리자에게 메시지를 보내고 클러스터를 할당받게 된다. 그렇지만, 하나의 관리자만을 사용하기 때문에 대규모 시스템에는 적합하지 않다.

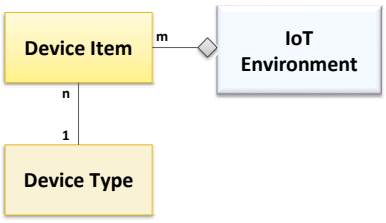
따라서, 본 논문에서는 IoT 컴퓨팅 환경에 존재하는 대량의 디바이스를 효율적으로 관리할 수 있도록 클러스터링 기법을 제안한다. 이를 위해 근접한 디바이스들과 관리 서버를 클러스터화하는 상세 수준의 설계를 제시하고, 제안한 기법을 구현 및 평가하여 기법의 효과를 확인한다.

3. 기반 연구

3.1 대규모 IoT 컴퓨팅 환경

IoT는 현재 많은 기업과 연구소에서 연구 중에 있으며, 2020년이면 약 260억개에 달하는 IoT 디바이스가 생

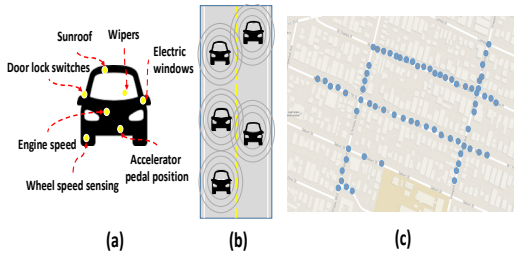
산될 것으로 예상된다[1]. IoT 디바이스의 수는 계속 증가하여 대규모 IoT 컴퓨팅 환경을 구축하게 될 것이다. 대규모 IoT 컴퓨팅 환경이란, 다양한 유형의 IoT 디바이스들이 많은 수의 인스턴스로 네트워크 상에 분산되어 디바이스간 협업하여 다양한 서비스를 제공하는 환경을 의미한다.



(그림 1) 대규모 IoT 컴퓨팅 환경을 이루는 요소간 관계
(Figure 1) Relationship for Elements for Large-Scale IoT Environment

(그림 1)과 같이 IoT 환경에는 m개의 디바이스가 존재하는데, 각 디바이스는 n 가지 유형을 가지게 된다. 즉 IoT 환경에는 n * m 디바이스들이 네트워크 영역 전체에 분산되어 존재하며, 서로 상호작용하며 사용자에게 각종 데이터 및 서비스를 제공하게 된다. 따라서, 네트워크 상에 분산되어 존재하는 n * m 개의 디바이스가 증가할 수록, 디바이스들을 관리하기 위한 비용은 기하급수적으로 증가하게 된다. 대규모 IoT 환경을 단일 서버를 이용하여 관리하는데 한계가 있기 때문에 디바이스들을 분산하여 관리할 필요성이 있다.

예를 들어, 스마트카(Smart Car)의 경우 다양한 유형의 디바이스 또는 센서들이 다수 내장되어 있다. 다음 (그림 2)는 대규모 IoT 환경에 대한 예제로서 스마트카와 관련된 그림들을 표현한다.



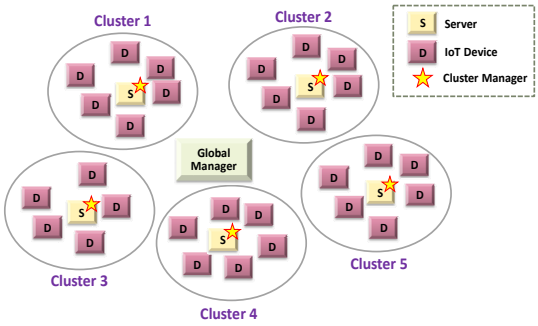
(그림 2) 대규모 IoT 환경에 대한 스마트카 예제
(Figure 2) Smart Car for Large-Scale IoT Environment

위 그림에서 (a)는 스마트카에 내장되어 있는 다양한 디바이스 또는 센서의 일부를 개념적으로 표현한다. 이와 같이 차량에 내장되어 있는 다수의 디바이스 및 센서들이 협업을 통해 (b)와 같이 도로상에서 다른 차량과 일정 거리를 유지 할 수 있도록 한다. (c)는 특정 도시의 도로를 나타내는데, 도로 위의 다수의 파란 점은 스마트카를 의미하며 이러한 자동차들이 도시내의 신호등이나 다른 디바이스들과의 협업을 통해 사용자들에게 다양한 서비스를 제공하게 된다[15][16][17].

본 논문에서는 위와 같은 대규모 IoT 환경을 오버헤드를 줄이면서 효율적으로 관리하기 위한 클러스터링 기법을 제안한다. 이 기법은 네트워크 상에 분산된 IoT 디바이스들을 일정 기준으로 클러스터화하여 각 클러스터 관리자와의 상호작용을 통해 디바이스를 모니터링하고 관리한다.

3.2 동적 클러스터링 기법

여러 문헌과 연구들에서 동적 클러스터링에 대한 연구가 진행 중이며, 본 절에서 동적 클러스터링을 IoT 컴퓨팅 환경에 적용한 경우에 대해 설명한다[4][5][6][13]. 대규모 IoT 컴퓨팅 환경에서의 클러스터링이란, IoT 디바이스와 애플리케이션을 일정한 기준에 의해 그룹화 하는 것을 의미한다. 다음 (그림 3)은 클러스터링 된 IoT 컴퓨팅 환경을 도식화하여 표현하고 있다.

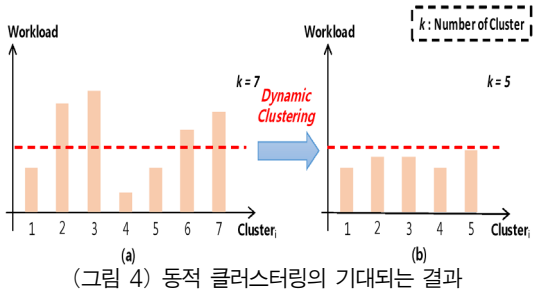


(그림 3) 클러스터링된 IoT 컴퓨팅 환경
(Figure 3) Overview of Clustered IoT Computing Environment

위와 같이 각각의 그룹들을 클러스터라고 부르며, 각 클러스터 내부에는 서버가 존재하고 이 서버가 클러스터 관리자(Cluster Manager)역할을 수행하며 노드인 IoT 디바이스를 관리한다. 그리고 이들 클러스터 관리자들은

전체 관리자(Global Manager)에 의해 관리된다. 지속적으로 변화하는 IoT 컴퓨팅 환경을 효율적으로 관리하기 위해서, 초기 설정되는 클러스터들을 계속 유지하는 것이 아니라, 실시간 평가를 통해 클러스터들을 변경하는 활동을 수행한다.

다음 (그림 4)는 동적 클러스터링을 통해 각 클러스터의 작업 부하를 균등하게 조절한 경우 예상되는 결과를 보여준다.



(그림 4) 동적 클러스터링의 기대되는 결과

(Figure 4) Expected Result of Dynamic Clustering

위 그림은 클러스터마다 처리해야 하는 작업량을 최초 클러스터링 되어 있는 (a)와 재클러스터링(Re-clustering)을 통해 작업량을 조절한 (b)로 나타내고 있다. 기존에 분할되어 있는 클러스터들을 실시간으로 평가한 결과를 바탕으로 재클러스터링하여 클러스터의 개수 k를 7에서 5로 줄였다. 작업이 상대적으로 적은 클러스터를 제거하고 부하가 많은 클러스터 내의 디바이스를 재배치함으로써 (b)와 같은 결과를 보이게 된다. 즉 k는 고정되어 있지 않으며, 실시간으로 클러스터 평가 후 필요한 경우에 재클러스터링을 통해 k를 조절할 수 있다.

IoT 컴퓨팅 환경에 동적 클러스터링을 적용하면 다음과 같은 장점들을 얻을 수 있다.

- **관리 오버헤드의 감소**
  (Minimizing Management Overheads)
  수시로 변화하는 IoT 컴퓨팅 환경을 효율적으로 관리하기 위한 클러스터를 계속 유지하기 때문에 관리를 위한 오버헤드를 낮은 상태로 유지
- **로드 밸런싱을 통한 작업 부하 조절**
  (Managing Overheads by Load Balancing)
  클러스터 내부의 IoT 디바이스들에서 발생하는 가변적인 부하를 상대적으로 작업량이 적은 클러스터로 재배치함으로써 최적화
- **에러 예방 및 전파 방지**
  (Preventing Errors and Their Propagation)
  클러스터 내부의 클러스터 관리자를 통해 각 IoT 디바이스가 모니터링되어 에러를 사전에 예방할 수 있고, IoT 디바이스들에서 문제가 발생하더라도 해당 클러스터 외부로 전파되는 것을 방지하고, 필요할 때 디바이스를 다른 클러스터로 변경하여 문제를 해결

본 논문에서 제안하는 동적 클러스터링 기법을 적용한 경우에, 전체 관리자에게 모든 정보를 취합하는 과정에서 정보의 일관성 유지 측면에서 추가 비용이 발생 할 수 있다. 이 문제를 해결하기 위해 전체 관리자를 구성하는 물리적 서버를 다수로 구성할 수 있고, 또 다른 방법으로 정보 취합을 위해 각 클러스터 관리자에 우선순위를 부여하여 중요도에 따라 정보 취합을 할 수 있다.

따라서, 본 논문에서는 동적 클러스터링 기법을 적용하여 IoT 컴퓨팅 환경이 변화함에 따라서 최적의 클러스터링을 구성하도록 하여 IoT 컴퓨팅 환경을 효율적으로 관리 할 수 있는 기법을 제안한다.

4. IoT 디바이스의 동적 클러스터링을 위한 정형 모델

IoT 컴퓨팅 환경을 동적 클러스터링하여 관리한다는 것은 유사한 위치에 존재하는 디바이스들을 하나의 클러스터로 정의하는 활동과 각 클러스터마다 고품질을 유지하는지 지속적으로 확인하는 활동의 결합이라고 할 수 있다. 유사한 위치의 디바이스들을 알기 위해서 디바이스들간의 논리적 근접도를 측정해야 하고, 하나의 클러스터가 고품질을 유지하는지 확인하기 위해 디바이스 관리 비용을 측정해야 한다. 논리적 근접도 측정을 위한 요소 목록 및 측정 함수는 클러스터링 연구를 수정하여 반영하였다[18].

4.1 IoT 디바이스의 논리적 근접도 측정을 위한 요소

논리적 근접도란 노드 사이의 물리적인 거리 외에 여러 기준을 이용하여 계산한 상대적인 가까움을 논리적으로 나타내는 척도를 의미한다[18]. 클러스터링 연구의 논리적 근접도 측정 수식을 이용하여 노드간 논리적 근접

도를 측정한다. IoT 컴퓨팅 환경을 클러스터링하기 위해서 먼저 모든 IoT 디바이스들간 논리적 근접도를 측정한다. 다음 (표 1)은 클러스터링 연구에서 논리적 근접도 측정을 위해 정의된 요소 중 IoT 컴퓨팅 환경에 맞는 요소 목록(Factor List)을 보여준다[18].

(표 1) 논리적 거리 함수를 위한 요소 리스트
(Table 1) Factor List for Logical Distance Function

DF Factor	Description	Unit
FLIST[1]	Geographical direct distance between two devices	Km
FLIST[2]	Frequency of invocations between two devices	
FLIST[3]	Average response time between two devices	Millisecond

논리적 요소뿐만 아니라 물리적 요소를 바탕으로 논리적 근접도를 판단하여 하나의 클러스터에 할당하게 되면, 두 노드간 상호작용 성능을 향상시킬 수 있다. 따라서 위의 (표 1)과 같이 논리적 근접도 측정 요소를 선정하였다. 본 논문에서는 IoT 컴퓨팅 환경을 클러스터링 하기 위해서는 최소한 위에 기술된 일반적인 기준 3가지를 고려하도록 한다. 만일 논리적인 거리에 영향을 주는 요소가 새롭게 발견되면 위 기준을 확장할 수 있다. 각 요소에 따라서 두 개의 IoT 디바이스간 논리적 근접도를 측정하게 되고, 모든 디바이스간 논리적 근접도를 측정한 이후에 근접한 디바이스들을 하나의 클러스터로 할당한다.

4.2 IoT 디바이스의 논리적 근접도 측정

클러스터링 된 IoT 컴퓨팅 환경에서는 클러스터 관리자 역할을 수행할 수 있는 노드는 컴퓨팅 파워를 가진 서버로서 그 수와 위치가 제한되어 있고, 이에 대해 클러스터링 되는 IoT 디바이스들이 영향을 받을 수밖에 없다. 이러한 영향을 최소화 하고 기존에 각종 서비스 제공을 위해 협업하고 있는 IoT 디바이스들에 본 논문에서 제안하는 기법을 적용할 수 있도록 디바이스들이 클러스터링 된 이후에 관리 노드를 각 클러스터에 할당하도록 한다. 따라서, IoT 디바이스간 논리적 근접도를 통해 서로 근접한 IoT 디바이스들을 동일한 클러스터에 할당하고, 그 이후에 IoT 디바이스와 관리 노드간 논리적 근접도 측정을 통해 각 클러스터와 가장 근접한 관리 노드를 할당한다.

논리적 근접도 측정에 사용되는 용어(Term)와 함수를 다음과 같이 정의한다.

- **DF:** 논리적 근접도 측정을 위한 함수를 나타낸다 (Distance Function). 이 값은 0부터 1까지의 범위를 가진다.
- **FLIST**: 논리적 근접도 측정을 위한 요소를 나타낸다 (Factor List).
- **numFLIST**: FLIST의 수를 나타낸다.
- **numProc(MGR$_i$, DEV$_j$)**: DEV$_j$ 로부터의 요청을 관리 노드가 처리한 수를 나타낸다.
- **numReq(MGR$_i$, DEV$_j$)**: IoT 디바이스로부터 관리 노드로 전달되는 요청의 수를 나타낸다.
- **JHC(MGR$_i$, DEV$_j$)**: IoT 디바이스가 관리 노드 MGRi로의 요청(Request)에 대한 처리(Proceeding) 결과를 나타낸다(Job Handling Capacity).
- **CLUSTER$_i$**: IoT 컴퓨팅 환경에 속한 i번째 클러스터를 의미한다.
- **numDev**: IoT 디바이스의 수를 의미한다.

다음은 두 디바이스간 논리적 근접도 측정을 위해 정의된 함수로서 4.1장의 FLIST 값을 이용하여 계산된다.

$$DF(DEV_i, DEV_j) = \frac{\sum_{i=1}^{numFLIST} FLIST[i] \times W_i}{numFLIST}$$

IoT 컴퓨팅 환경은 특정 도메인에 제한된 디바이스들을 관리하기 위한 목적이 아니기 때문에 각 요소마다 가중치가 달라질 수 있다. 예를 들어, 스마트홈 환경에서는 물리적인 거리가 다 근거리이기 때문에 FLIST[1]의 Wi를 다른 요소에 비해 낮게 책정하여 논리적 근접도를 구하도록 한다.

관리 노드의 수는 최소 하나에서 복수가 될 수 있다. 이 때 각각의 관리 노드가 어떤 클러스터를 관리 할지 정하기 위해서 특정 클러스터 내부의 모든 디바이스에 대하여 관리 노드간 작업 처리량을 구하는 방식을 이용한다. 다음은 하나의 디바이스로부터 관리 노드로의 요청을 해당 관리 노드가 얼마나 처리할 수 있는지를 나타내는 함수이다.

$$JHC(MGR_i, DEV_j) = \frac{numProc}{numReq}$$

다음은 특정 클러스터 내부의 모든 디바이스들로부터 관리 노드로의 작업 처리량의 평균을 구하는 함수이다. 이를 통해 구해진 DF가 0에 가까울수록 논리적으로 가까움을 나타내기 때문에 최소의 DF를 가진 관리 노드를 해당 클러스터를 관리하도록 한다.

$$DF(MGR_i, DEV_j) = \frac{\sum_{DEV_j \in SetDevs(CLUSTER_i)} JHC(MGR_i, DEV_j)}{numDev}$$

위 함수에 의해서 모든 디바이스간 논리적인 거리가 측정되면, 각 클러스터가 정의되고 이후에 새로운 디바이스가 추가되거나 재클러스터링이 필요한 경우에 위 함수들에 의해서 다시 논리적 근접도를 측정하게 된다.

4.3 IoT 디바이스의 관리 비용 측정을 위한 요소

IoT 컴퓨팅 환경에서의 IoT 디바이스는 이동성을 갖거나 다양한 네트워크 프로토콜을 사용한다는 특징이 있다. 이러한 요소들은 논리적 근접도 측정에 직접적인 영향을 주지는 않지만 추가적인 관리에 대한 비용을 증가시켜 IoT 컴퓨팅 환경의 클러스터링 품질에 영향을 줄 수 있다. 예를 들어, 클러스터링이 완료된 IoT 디바이스들이라도 이동성에 의해서 소속된 클러스터를 벗어나게 된다면 품질을 떨어뜨리거나, 중계 노드가 필요한 ZigBee를 사용하는 디바이스의 경우 중계 노드로 인해 추가로 관리 비용이 증가함에 따라서 전체 클러스터 품질에 영향을 줄 수 있다. 따라서, 이러한 요소들을 관리 비용 요소 목록으로 정의한다.

본 논문에 적용된 관리 비용에 영향을 줄 수 있는 요소들은 다음 (표 2)와 같다.

(표 2) 동적 클러스터링에 대한 관리 비용 측정 요소
(Table 2) Management Cost Factor for Dynamic Clustering

Management Cost Factor	Description
MCFLIST[1]	Check whether or not an IoT device has mobility
MCFLIST[2]	Check whether or not an IoT device needs a broadcasting node for network protocols (Bluetooth, WiFi, ZigBee)

IoT 디바이스의 유형 중에는 이동성을 가진 디바이스가 존재한다. 한곳에 고정되어 센싱 및 액츄에이션을 하지 않고, 상황에 따라서 자동 혹은 사용자에 의해서 위치를 바꿔가며 디바이스의 목적을 수행한다.

다양한 네트워크 프로토콜 중에서 (표 2)의 일반적으로 IoT 디바이스에 사용되는 대표 프로토콜 3가지를 고려한다. 각 프로토콜의 특성 때문에 직접 관리 노드와의 상호작용이 어려운 디바이스가 존재할 수 있다. 클러스터링에 영향을 주는 프로토콜마다의 특성은 다음과 같다.

- **Bluetooth**: 거리에 따라 다름. 관리 노드와 커뮤니케이션 범위(약 1~100m) 이내면 직접 통신 가능 하지만, 범위 밖이면 중계 노드 필요함.
- **WiFi**: 관리 노드와 직접 통신 가능함.
- **ZigBee**: 관리 노드와 디바이스간 통신을 위해서 중계 노드 필요함.

이를 해결하기 위해서 디바이스와 관리 노드를 연결해주는 중계 노드를 활용하게 되는데, 관리 노드에서 디바이스 관리를 위해 중계 노드를 거쳐야 하는 오버헤드가 발생하게 되어 관리 비용이 증가하게 된다. IoT 컴퓨팅 클러스터링 환경에서 관리 비용 측정을 위해 최소한 위 요소 목록을 고려하도록 한다. 만일 관리 비용에 영향을 주는 요소가 새롭게 발견되면 위 기준을 확장할 수 있다.

4.4 IoT 디바이스의 관리 비용 측정

IoT 디바이스가 이동성을 가지고 있는 경우에는 클러스터링 품질을 평가 할 때 해당 디바이스를 지속적으로 확인해야 하고, 필요한 경우에 클러스터 변경을 해주는 등의 추가적인 활동이 필요하기 때문에 관리 비용이 증가하게 된다.

IoT 디바이스가 중계 노드를 사용할 경우에는 클러스터링의 품질 평가시 해당 IoT 디바이스 이외에 중계 노드까지 확인하는 활동이 필요하다. 따라서, 추가 활동 여부에 따라서 0 또는 1의 값을 갖도록 한다. 사용할 수 있는 네트워크 프로토콜이 Bluetooth, WiFi, ZigBee 중에서 Bluetooth는 환경에 따라 중계 노드 사용 여부가 결정되고, ZigBee는 중계 노드 없이는 관리 노드와 통신이 어렵기 때문에, 이러한 프로토콜을 사용하고 있는 디바이스들은 관리 비용이 추가적으로 발생하게 된다.

관리 비용 측정에 사용되는 용어는 다음과 같이 정의한다.

- **MCFLIST**: 관리 비용 요소 목록(Management Cost Factor List)으로서 디바이스 관리에 영향을 끼치는 요소를 의미한다.
- **MCF(MGR$_i$, DEV$_j$)**: 관리 노드 MGRi와 IoT 디바이스 DEVj간 관리 비용을 측정하기 위한 함수를 의미한다 (Management Cost Function).

다음은 위 MCFLIST 측정을 위해 정의된 함수로서 관리 노드와 디바이스의 특성에 따라 요소에 값을 부여한다.

$$MCF(MGR_i, DEV_j) = \frac{\sum_{i=1}^{numMCFLIST} MCFLIST[i] \times W_i}{numMCFLIST}$$

위 함수는 MGRi와 DEVj간의 MCFLIST의 평균치를 나타내며 0부터 1의 값을 갖는다.

다음은 MCF를 통해 측정된 관리 노드와 클러스터 내의 모든 디바이스간 관리 비용을 계산하는 함수이다.

$$MC(CLUSTER_i) = \sum_{\substack{MGR_i \in SetMGR(CLUSTER_i), \\ DEV_j \in SetDevs(CLUSTER_i)}} MCF(MGR_i, DEV_j)$$

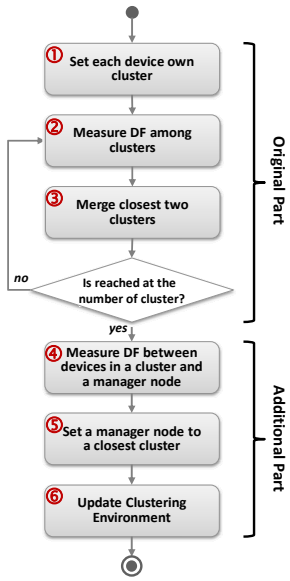
위 함수는 IoT 컴퓨팅 환경의 전체 클러스터링 품질을 평가할 때 영향을 준다.

5. IoT 컴퓨팅을 위한 동적 클러스터링 서비스 구현

5.1 IoT 특화된 Hierarchical 클러스터링 알고리즘

IoT 컴퓨팅 환경은 대량의 IoT 디바이스들이 네트워크 상에 존재하는 환경이다. 따라서, 이러한 환경을 클러스터링 하는 알고리즘의 입력값으로 대량의 IoT 디바이스가 되는데, 입력값이 크기 때문에 클러스터링에 드는 부하가 증가할 수 있다. Hierarchical 클러스터링 알고리즘은 입력값인 대량의 디바이스들에 직접 접근하여 클러스터링 하지 않고 각 디바이스들로부터 계산되는 근접도를 바탕으로 클러스터링 되기 때문에 클러스터링에 대한 부하가 상대적으로 작다[19][20].

본 논문에서 사용하는 Hierarchical 클러스터링 알고리즘은 IoT 컴퓨팅 환경의 특성 때문에 일부 수정하여 사용한다. 아래 (그림 5)는 IoT에 특화된 Hierarchical 클러스터링 알고리즘을 나타낸다.



(그림 5) 특화된 Hierarchical 클러스터링 알고리즘
(Figure 5) Specialized Hierarchical Clustering Algorithm

위 그림은 기존 Hierarchical 클러스터링 알고리즘 부분인 ①~③에 ④~⑥활동을 추가하여 알고리즘을 표현하고 있다. 클러스터링 전에 입력값으로 몇 개의 클러스터로 IoT 컴퓨팅 환경을 구성할지 입력해야 한다. 이 입력값은 관리 노드의 수와 관련이 있는데 최소 1부터 최대 네트워크 상에 존재하는 관리 노드의 수까지 입력 가능하다. ①에서는 네트워크 상에 존재하는 IoT 디바이스들을 각각 하나의 클러스터로 지정하는 활동이다. ②에서 각 클러스터 내부에 존재하는 디바이스들간 근접도의 평균을 구한다. 각 클러스터마다 최소 근접도를 가지고 있는 디바이스를 기준으로 클러스터간 근접도를 구하게 된다. ③에서 ②에서 구한 근접도를 바탕으로 가장 근접한 클러스터들을 병합한다. 이후에 처음에 입력 받은 클러스터의 수와 현재 클러스터 수를 비교하여 두 수가 같아질 때까지 ②와 ③ 활동을 반복해서 수행한다. ④에서 각 클러스터 내부의 디바이스들과 관리 노드들간의 논리적 근접도를 구하는데, 이를 바탕으로 논리적으로 가장 가까운 관리 노드와 클러스터를 식별하게 된다. ⑤에서 ④에서 구해진 클러스터와 관리 노드간 근접도를 통해서 각 관리 노드를 가장 근접한 클러스터를 관리하도록 할당한다. ⑥에서 지금까지 구성된 클러스터링 환경을 저장하여 전체 클러스터링 활동을 종료 시킨다.

동적 환경에서 관리 노드를 선정 및 제거하기 위해서 클러스터 내부 노드간 논리적 근접도를 의미하는 DFTable 을 사용하게 된다. 클러스터 내에서 다른 노드들간 거리가 평균적으로 가장 가까운 노드를 클러스터 관리 노드로 선정하게 된다.

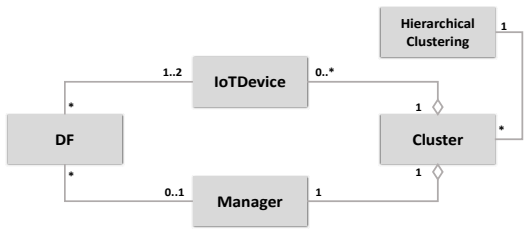
(표 3) 클러스터 관리 노드 선정에 대한 의사 코드
(Table 3) Pseudo Code for Selecting Cluster Manager Node

```
1 Begin
2 // Set initial nodes
3 Enter a set of node, NODE = {x1, x2, …,
4  xn}.
5 // Measure DF among nodes in a cluster
6  DF_TABLE = makeDFTable(NODE);
7 // Repeat Setting Cluster Manager
8 Repeat until number of Clusters
9   Repeat until finding the node
10   // Find the closest node
11     n = findMinimumNode(CLUSTERi);
12      // Set the node to a manager node
13     setManagerNode(n);
14 // Update IoT Computing Env.
15 updateEnv(CLUSTERS);
16 End
```

위 의사 코드에서 8~13라인을 통해서 각 클러스터 내부의 논리적 거리를 바탕으로 관리 노드의 선정 및 변경이 이루어진다. 논리적 거리는 IoT컴퓨팅 환경에서 동적으로 측정되어 반영된다.

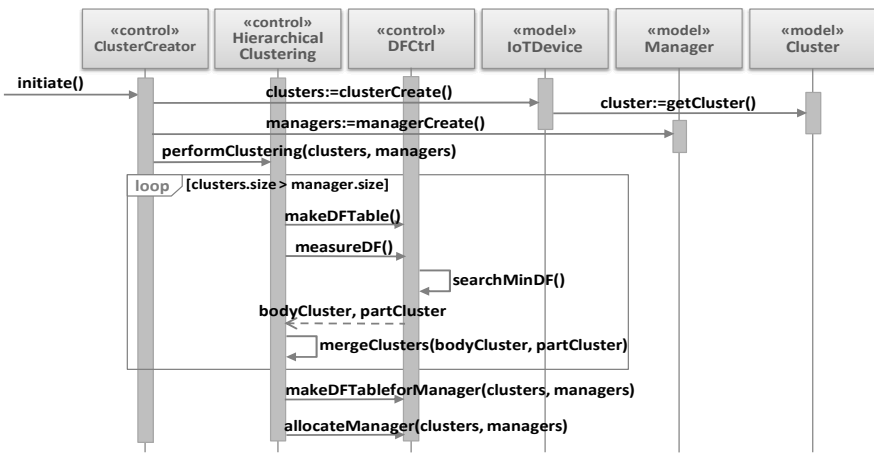
5.2 동적 클러스터링 서비스 설계 모델

다음 (그림 6)는 동적 클러스터링에 참여하는 클래스들간 관계를 개념적 객체 모델링을 통해 보여준다.



(그림 6) 동적 클러스터링에 대한 클래스 다이어그램
(Figure 6) Class Diagram for Dynamic Clustering

본 논문에서 제안한 Hierarchical 클러스터링 알고리즘은 *Cluster*에 적용되어 모든 디바이스들을 클러스터링 하게 된다. 최초 클러스터링 진행 전에는 *Cluster* 하나당 *IoTDevice* 하나만 할당된 상태이며, 클러스터링이 종료되면 *Cluster*에는 한 개 이상의 *IoTDevice*와 *Manager*가 존재한다. 클러스터링은 두 개의 *IoTDevice*간 논리적 근접도인 DF와 *IoTDevice*와 *Manager*간 DF를 구해서 근접한 노드들을 하나의 클러스터로 구성한다.



(그림 7) 동적 클러스터링에 대한 시퀀스 다이어그램
(Figure 7) Sequence Diagram for Dynamic Clustering

위 (그림 7)은 IoT에 특화된 Hierarchical 알고리즘의 동적 모델링에 대한 다이어그램으로서 IoT 디바이스들간 논리적 근접도 측정부터 각 클러스터에 관리 노드를 할당하는 부분까지 보여준다.

본 논문에서 제안하는 클러스터링에는 총 6개의 클래스가 참여한다. *ClusterCreator*가 IoT 컴퓨팅 환경의 형상관리 저장소에 접근하여 클러스터링에 필요한 *IoTDevice*와 *Manager*, *Cluster*를 초기화 하는 역할을 담당한다. *HierarchicalClustering*은 클러스터링 알고리즘의 전체 흐름을 관리하여 IoT 컴퓨팅 환경을 클러스터링 한다. 이때 *DFCtrl*이 디바이스간 논리적 근접도를 측정하고 전체 클러스터간 가장 근접한 디바이스들을 찾아내어 해당 디바이스가 포함된 두 개의 클러스터를 찾는다. 탐색된 두 클러스터를 *HierarchicalClustering*에서 병합하고 전체 디바이스들이 클러스터마다 할당 되면 클러스터와 가장 근접한 *Manager*를 찾고 각 클러스터를 관리하도록 관리자 노드를 지정한다.

5.3 동적 클러스터링 서비스 구현 결과

클러스터링을 적용하기 위해서는 대량의 IoT 디바이스와 이러한 디바이스들을 관리하기 위한 관리 노드들이 필요하다.

클러스터링 알고리즘이 적용된 IoT 컴퓨팅 환경은 IoT 디바이스 100개를 10개의 클러스터로 분할하고 각 클러스터를 10개의 관리자 노드에 의해 관리하도록 한다. 실제 IoT 환경 운영시에는 IoT 디바이스들간 논리적 근접도를 4. 장의 디바이스들간 상호 작용 등을 모니터링하여 계산된다. 다음 (그림 8)은 IoT 디바이스 100개에 대해 Hierarchical 클러스터링을 적용한 결과를 보여준다.

```
[Cluster Manager Name: M.9l] QoCluster: 0.0254
(DEV1)(DEV50)(DEV98)(DEV25)(DEV71)(DEV79)(DEV91)(DEV31)(DEV34)(DEV83)

[Cluster Manager Name: M.te] QoCluster: 0.0208
(DEV2)(DEV20)(DEV23)(DEV26)(DEV81)(DEV96)(DEV14)(DEV66)(DEV85)

[Cluster Manager Name: M.a1] QoCluster: 0.0097
(DEV3)(DEV7)(DEV8)(DEV10)(DEV76)(DEV12)(DEV27)(DEV16)(DEV99)(DEV46)(DEV63)(DEV97)

[Cluster Manager Name: M.46] QoCluster: 0.0322
(DEV4)(DEV21)(DEV59)(DEV33)(DEV70)(DEV60)(DEV86)

[Cluster Manager Name: M.eh] QoCluster: 0.0271
(DEV5)(DEV48)(DEV95)(DEV17)(DEV52)(DEV22)(DEV72)(DEV82)(DEV100)(DEV62)(DEV92)

[Cluster Manager Name: M.bj] QoCluster: 0.0057
(DEV6)(DEV9)(DEV11)(DEV13)(DEV29)(DEV15)(DEV78)(DEV89)(DEV35)(DEV42)(DEV90)(DEV36)(DEV88)(DEV54)

[Cluster Manager Name: M.gb] QoCluster: 0.0676
(DEV18)(DEV56)(DEV19)(DEV28)(DEV30)(DEV32)(DEV49)(DEV65)(DEV68)(DEV75)

[Cluster Manager Name: M.9m] QoCluster: 0.0411
(DEV24)(DEV38)(DEV47)(DEV41)(DEV73)(DEV84)(DEV53)(DEV80)(DEV94)(DEV55)(DEV64)(DEV74)

[Cluster Manager Name: M.me] QoCluster: 0.0083
(DEV37)(DEV45)(DEV39)(DEV61)(DEV40)(DEV67)(DEV44)(DEV58)

[Cluster Manager Name: M.sw] QoCluster: 0.0061
(DEV43)(DEV57)(DEV51)(DEV69)(DEV77)(DEV87)
```

(그림 8) 클러스터링 결과 예제
(Figure 8) Example Result for Clustering

위 그림은 10개의 클러스터마다 관리자 노드가 할당되어 있고, 클러스터에 디바이스가 추가될수록 클러스터의 범위가 늘어나는데, 이에 따라서 범위가 큰 클러스터에 근접한 디바이스들이 점차 늘어나게 되어 클러스터링 환경 내의 대부분의 디바이스가 할당되는 경향이 있다. 따라서, 각 클러스터 내의 디바이스들이 특정 클러스터에만 집중되지 않도록 다음 (표 4)과 같이 구현하여 적용하였다.

(표 4) 클러스터링에 대한 소스 코드
(Table 4) Code Snippet for Clustering

```
1 private void
2 searchMinDF(CustomArrayList<Cluster>
3 clusters){
4   // Search Minimum DF
5   for(int i=0; i<DFTable.length; i++) {
6       for(int j=0; j<DFTable.length;j++) {
7           if (i == j)
8              continue;
9           else {
10      if (DFTable[i][j].value<MIN_DF &&
11          DFTable[i][j].value>SEC_MIN_DF) {
12        MIN_DF = DFTable[i][j].value;
13          bodyClusterIndex = i;
14          partClusterIndex = j;
15      }
16     }
17    }
18  }
19  }
20  }
```

위 searchMinDF 메소드는 현재 구성되어 있는 클러스터간 가장 근접한 클러스터의 식별 지표(index)를 찾는 역할을 수행한다. 8~12 라인을 통해 현재 최소인 DF를 찾는데, 만약 현재 탐색된 클러스터에 포함되어 있는 디바이스의 수가 전체 평균보다 클 경우가 있을 수 있다. 이를 해결하기 위해 8~9 라인에서 이전에 탐색된 최소 DF보다는 크지만 최소인 DF, 즉 논리적 근접도 테이블에서 최소 DF보다 두 번째로 작은 DF를 찾도록 한다.

6. 실험 및 평가

6.1 IoT 컴퓨팅 환경의 최적화 평가 모델

동적 클러스터링은 IoT 컴퓨팅 환경의 관리 품질에 중대한 영향을 미치기 때문에 클러스터링이 효율적으로 이루어졌는지 평가 및 확인해야 한다. IoT 컴퓨팅 환경의

최적화 정도를 평가하기 위해 정량적 측정이 가능한 품질 메트릭을 본 장에서 정의한다. 다음과 같이 클러스터링의 품질 측정 메트릭에 사용되는 용어와 함수를 정의한다.

- ***IoTEnv***: IoT 컴퓨팅 환경을 나타낸다.
- ***CLUSTER$_i$***: *IoTEnv*에 속한 *i* 번째 클러스터를 나타낸다.
- ***NumClusters***: *IoTEnv*를 구성하는 모든 클러스터의 개수를 나타낸다.
- ***DEV$_i$***: IoTEnv에 참여하는 *i* 번째 IoT 디바이스를 나타낸다.
- ***MGR_NODE$_i$***: 클러스터 내의 디바이스들을 관리하는 *i*번째 *CLUSTER$_i$*의 관리자 노드를 나타낸다.
- **NumDEVS**: IoTEnv에 속한 모든 디바이스들과 관리자 노드의 개수이다. 즉, IoTEnv = {MGR_NODE$_1$, DEV$_1$, DEV$_2$, …, DEV$_{NumDEVS}$} 이다.
- ***SizeCluster(CLUSTER$_i$)***: 특정 클러스터 *CLUSTER$_i$*에 속한 디바이스들과 관리자 노드의 개수를 반환하는 함수이다.
- ***SizeManager(CLUSTER$_i$)***: 특정 클러스터 *CLUSTER$_i$*에 속한 관리자 노드의 개수를 반환하는 함수이다. 단, 본 논문에서는 클러스터마다 한 개의 관리자를 가진다.
- ***SetDEVS(CLUSTER$_i$)***: 특정 클러스터 *CLUSTER$_i$*에 속한 모든 디바이스들 집합을 반환하는 함수이다.
- ***SetMGR(CLUSTER$_i$)***: 특정 클러스터 *CLUSTER$_i$*에 속한 관리자 노드를 반환하는 함수이다.
- ***DF(DEV$_i$, DEV$_j$):*** 두 개의 IoT 디바이스 *DEV$_i$*와 *DEV$_j$* 사이의 논리적 근접도를 계산하는 함수이다.
- ***DF(MGR$_i$, DEV$_j$):*** 하나의 관리자 노드와 디바이스 사이의 논리적 근접도를 계산하는 함수이다.
- ***MC(CLUSTER$_i$):*** *MCF*를 이용해 *CLUSTER$_i$* 내의 모든 디바이스와 관리 노드간 관리 비용을 계산하는 함수이다.

*IoTEnv*의 최적화 정도는 환경 내의 각 클러스터가 최적으로 구성되었는지와 전체 환경이 최적으로 구성되었는지를 평가함으로써 알 수 있다.

클러스터의 최적화 정도는 해당 클러스터내의 디바이스와 관리자 노드가 높은 응집도를 가지고 있는지 평가한다. 이를 위해 다음과 같은 클러스터 최적화 측정 함수를 정의한다.

$$QoCluster(CLUSTER_i) = \frac{\sum_{DEV_x, DEV_y \in SetDevs(CLUSTER_i)} DF(DEV_x, DEV_y)}{SizeCluster(CLUSTER_i) - SizeManager(CLUSTER_i)} + \sum_{MGR_i \in SetMGR(CLUSTER_i), DEV_x \in SetDevs(CLUSTER_i)} DF(MGR_i, DEV_x)$$

위 함수에서는 디바이스들 간 응집도와 디바이스들과 관리자 노드간 응집도를 측정한다. *QoCluster(CLUSTER$_i$)*는 0과 1사이의 값을 가진다. 논리적으로 근접한 디바이스들과 이러한 디바이스들에 근접한 관리자 노드일수록 0에 더욱 근접한 값을 반환한다.

*IoTEnv*의 환경의 최적화 정도를 측정하기 위해서 현재 구성되어 있는 클러스터들이 최적화 되어 있는지(상호간 결합도가 낮은지)를 평가한다. 이는 클러스터간에 상호작용이 적은 대신에 디바이스 내의 관리자 노드와 상호작용이 많다는 것을 의미하며, 위와 같은 클러스터가 많을수록 *IoTEnv*의 전체 품질이 높아진다. 이러한 *IoTEnv* 전체 품질 측정을 위해 다음과 같은 함수를 정의한다.

$$QoIoTEnv = \frac{\sum_{i=1}^{NumClusters} QoCluster(CLUSTER_i) + MC(CLUSTER_i)}{NumClusters}$$

위와 같이 *IoTEnv* 수준의 최적화 정도는 0과 1사이의 값을 반환한다. 클러스터간 상호작용이 적고 관리자 노드와 상호작용이 많을 때 0에 근접한 값을 반환한다.

6.2 실험 환경

실제 IoT 컴퓨팅 환경을 구축하기 위한 IoT 디바이스 및 관리 서버 확보에 대한 비용이 크기 때문에 클러스터링 기법을 시뮬레이션 환경에 적용하여 결과 도출 및 평가한다. 실험 시뮬레이션은 Windows 7에서 Java7과 Eclipse로 구현하였고, (표 5)와 같은 실험 환경에 대해 진행 하였다.

(표 5) 실험을 위한 최초 IoT 컴퓨팅 환경
(Table 5) Initial IoT Computing Environment for Experiments

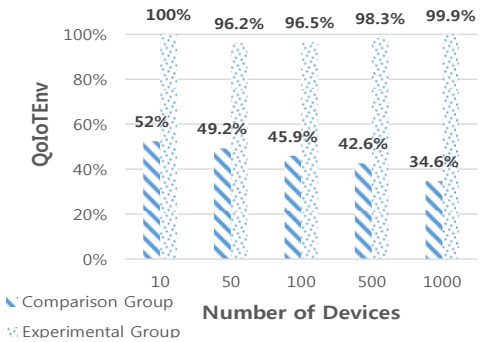
Experiments \ Factors			Number of IoT devices	Number of Clusters
Type #1	Comparison Group	Case 1	10	1
		Case 2	50	
		Case 3	100	
		Case 4	500	
		Case 5	1000	

Experiments \ Factors			Number of IoT devices	Number of Clusters
Experimental Group		Case 1	10	10
		Case 2	50	
		Case 3	100	
		Case 4	500	
		Case 5	1000	
Type #2		Case 1	100	1
		Case 2		5
		Case 3		10
		Case 4		50
		Case 5		100

위 두 가지 유형의 실험을 통해 클러스터링의 효과와 클러스터의 수에 대한 클러스터링의 효율을 측정한다. 첫 번째인 Type #1은 클러스터링이 적용되지 않은 환경인 대조군과 클러스터링이 적용된 환경인 실험군을 비교하기 위한 실험이다. 두 번째인 Type #2은 IoT 디바이스의 수를 100개로 고정하고, 관리자 노드의 수를 1부터 100까지 변경하며 총 5가지 유형에 대한 클러스터링 환경을 구성하여 클러스터의 수가 증가함에 따른 클러스터링의 효율을 확인한다.

6.3 실험 결과 및 평가

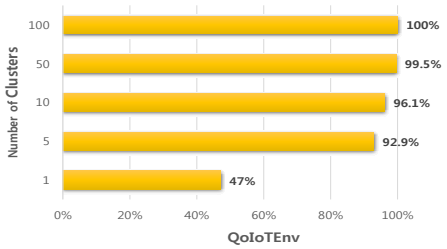
클러스터링의 효과를 측정하기 위한 첫 번째 실험에서 클러스터링을 적용 여부에 대한 결과를 대조군과 실험군의 총 10가지 유형에 대해서 20회씩 반복하여 (그림 9)과 같이 평균치를 측정하였다.



(그림 9) 클러스터링 적용에 대한 비교표
(Figure 9) Comparison between WITH and W/O Clustering

위 실험에서는 측정된 *QoIoTEnv*를 백분율로 환산하여 나타냈는데, 100%에 가까울수록 클러스터링 적용시 효과가 크다. 클러스터의 수가 한 개인 대조군은 디바이스의 수가 10에서 1000까지 늘어나면서 효과가 감소하고 있는데, 이는 관리 노드가 관리해야 할 디바이스의 수가 늘어남에 따른 오버헤드 증가가 주요 원인이다. 반면, 클러스터가 10개인 실험군에서는 디바이스의 수가 증가하더라도 90% 이상의 *QoIoTEnv*를 보이면서 클러스터링이 효과적으로 적용되었다. 특히, 실험군에서 디바이스의 수가 10개인 경우 *QoIoTEnv*가 100%라는 수치가 측정되었다. 1개의 디바이스를 1개의 관리 노드가 관리한다 것이고, 평가 모델에서 DF 이외의 외부 요소에 대한 영향은 차단한 상태에서 평가가 진행되었기 때문에 나타난 결과이다.

클러스터의 수에 대한 클러스터링의 효율을 측정하기 위한 두 번째 실험에서 총 5가지 유형에 대해 20회씩 반복하여 평균치를 측정하였다. 다음 (그림 10)은 두 번째 실험에 대해 측정된 *QoIoTEnv*를 백분율로 환산하여 나타냈다. 실험 결과가 100%에 가까울수록 클러스터링이 효율적이다.



(그림 10) 클러스터의 수에 따른 QoIoTEnv의 관계 비교
(Figure 10) Relationship between Number of Clusters and QoIoTEnv

위 실험에서는 디바이스의 수가 100개로 고정이 된 환경에서 클러스터의 수만 1부터 100까지 증가시켰다. 클러스터의 수가 1일 때 47%의 *QoIoTEnv*가 클러스터의 수가 5로 증가하면서 92.9%로 약 2배 이상 증가한 수치를 보여주고 있다. 이는 관리 노드가 증가함으로 인해 처리해야 하는 디바이스가 효과적으로 분포되어 효율이 증가했다. 그렇지만 클러스터의 수를 5개 이상 100개까지 증가 시키게 되면 *QoIoTEnv*가 4~5% 내외의 수치만 증가한다. 이는 디바이스의 수에 따라서 최대 효과를 얻을 수 있는 관리 노드의 수가 존재함을 의미한다. 위 실험에서는 100개의 디바이스 관리를 위해서 5개 이상의 관리 노드를 추가한다면 클러스터링의 최대 효과를 얻기 힘들다.

두 가지 실험을 통해서 대량의 IoT 디바이스들을 관리하기 위해 IoT 컴퓨팅 환경을 클러스터링 하고, 최적의 클러스터를 찾아서 관리 노드를 배치한다면 IoT 디바이스 관리에 대해 최소의 비용으로 최대 효과를 얻을 수 있음을 확인했다.

7. 결 론

IoT 디바이스에 대해서 다양한 제품들이 생산되고 있지만, 이러한 디바이스들이 존재하게 되는 IoT 컴퓨팅 환경과 관련된 연구는 아직 초기 단계에 있다.

IoT 컴퓨팅 환경 내에 대량의 IoT 디바이스가 동시에 존재하게 되는데, 이때 이들 디바이스들의 모니터링 및 제어를 위한 관리 노드인 서버가 필요하다. 그렇지만 대량의 디바이스들을 하나의 서버를 이용해서 관리한다면 그에 따른 오버헤드 때문에 서버는 효과적으로 디바이스들을 관리하기 어렵다. 그래서 디바이스 관리를 위한 서버를 한대 이상 IoT 컴퓨팅 환경에 배치하여 디바이스를 관리하려고 하는데, 이때 어떤 관리 노드를 어디에 몇 대를 배치해야 하는지 알기 어렵다.

본 논문에서는 기반 연구로서 대규모 IoT 컴퓨팅 환경과 동적 클러스터링 기법에 대한 개념을 소개하였다. 그리고 IoT 컴퓨팅 환경에 존재하는 디바이스들을 관리하는데 드는 비용을 줄이기 위해 클러스터링을 하게 되는데, 이 때 디바이스간 거리가 얼마나 가까운지를 측정하기 요소로서 논리적 근접도를 소개했다. 이러한 논리적 근접도를 바탕으로 Hierarchical 알고리즘을 IoT 컴퓨팅 환경에 맞게 특화 하여 동적 클러스터링 기법을 설계하였다. 설계된 기법이 IoT 컴퓨팅 환경에 적용했을 때 클러스터링 결과가 최적으로 진행되었는지 확인하기 위해 평가 모델을 정의했고, 이를 바탕으로 각 클러스터의 품질 및 전체 클러스터링 환경을 평가하는 실험을 진행하여, 제안된 기법의 효과와 적용성을 확인하였다. 따라서, 본 논문에서 제안한 기법을 IoT 컴퓨팅 환경에 적용하면 대량의 IoT 디바이스들을 최적의 클러스터에 배치하여 저비용 고효율로 관리 할 수 있게 된다.

참 고 문 헌 (Reference)

[1] Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020 [Internet], http://www.gartner.com/newsroom/id/2636073

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Journal on Future Generation Computer Systems*, Vol. 29, No. 7, pp. 1645-1660, 2013.

[3] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelffle, “Vision and challenges for realizing the Internet of Things,” *European Commission - Information Society and Media DG*, Luxembourg, 2010.

[4] N. Nasser, L. Karim, A. Ali, and N. Khelifi, “Multiple Base Station and Packet Priority-based Clustering Scheme in Internet of Things,” *in Proceedings of the 2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pp. 58-61, Da Nang, 2014.

[5] P. Krishan, “A Study on Dynamic and Static Clustering Based Routing Schemes for Wireless Sensor Networks,” *International Journal of Modern Engineering Research (IJMER)*, Vol. 3, No. 2, pp. 1100-1104, 2013.

[6] Y. Zhang, L. T. Yang, and J. Chen, “Clustering in Wireless Sensor Networks,” in *RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations*, CRC Press, ch. 12, 2009.

[7] L. Ding, P. Shi, and B. Lie, “The Clustering of Internet, Internet of Things and Social Network,” *in Proceedings of the 2010 3rd International Symposium on Knowledge Acquisition and Modeling*, pp. 417-420, 2010.

[8] V. Gazis, K. Sasloglou, N. Frangiadakis, and P. Kikiras, “Wireless Sensor Networking, Automation Technologies and Machine to Machine Developments on the Path to the Internet of Things,” *in Proceedings of the IEEE 2013 16th Panhellenic Conference on Informatics*, pp. 276-282, 2012.

[9] L. Zhang, A. S. Atkins, and H. Yu, “Knowledge Management Application of Internet of Things in Construction Waste Logistics with RFID Technology,” *Journal of Computing Science and Communication Technologies*, Vol. 5, No. 1, pp. 760-767, 2012.

[10] Smart Santander [Internet], http://www.smartsantander.eu

[11] Internet of Things (IoT) for Creating Smart Cities [Internet], http://issnip.unimelb.edu.au/research_program/Internet_of_Things

[12] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From Today's Intranet of Things to a Future Internet of Things: A Wireless- and Mobility- Related View," *Journal on Wireless Communications*, Vol. 17, No. 6, pp. 44-51, 2010.

[13] M. Gerla and J.T.C. Tsai, "Multicluster, mobile, multimedia radio network, Wireless Networks," *Wireless Networks,* Vol. 1, pp. 255-265. 1995.

[14] M. Du, X. Wang, D. Wang, and Y. Wang, "Device-to-Device Dynamic Clustering Algorithm in Multicast Communication," *in Proceedings of the 2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, Chengdu, pp. 578-582, 2013.

[15] Y. Zheng, S. Rjasegarar, C. Leckie, and M. Palaniswami, "Smart Car Parking: Temporal Clustering and Anomaly Detection in Urban Car Parking," *in Proceedings of the 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1-6, Singapore, 2014.

[16] C. Wang, Z. Shen, E. Tian, and Q. Zheng, "Multi-smart car control system design and research based on ZigBee," *in Proceedings of the 26th Chinese Control and Decision Conference (2014 CCDC)*, pp. 1490-1494, Changsha, 2014.

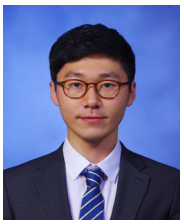
[17] J. Sun Y. Zhang, and J. Fan, "SmartAgents: A Scalable Infrastructure for Smart Car," in Proceedings of the 2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 99-103, Gwangju, 2011.

[18] D. Y. Kim, H. J. La, and S. D. Kim, "Dynamic Clustering based Optimization Technique and Quality Assessment Model of Mobile Cloud Computing," *KIPS Transactions on Software and Data Engineering*, Vol. 2, No. 6, pp. 383-394, 2013.

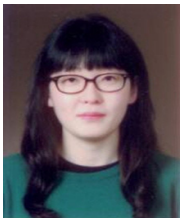
[19] M. K. Rafsanjani, Z. A. Varzaneh, and N. E. Chukanlo, "A survey of hierarchical clustering algorithms," *Journal on Mathematics and Computer Science*, Vol. 5, No. 3, pp. 229-240, 2012.

[20] P. N. Mahalle, N. R. Prasad, and R. Prasad, "Novel Context-aware Clustering with Hierarchical Addressing (CCHA) for the Internet of Things (IoT)," *in Proceedings of the 5th International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013)*, pp. 267-274, Bangalore, 2013.

◐ 저 자 소 개 ◑



김 대 영 (Dae Young Kim)
2011년 성공회대학교 소프트웨어공학과 졸업 (학사)
2014년 숭실대학교 컴퓨터학과 졸업 (석사)
관심분야: 사물인터넷 컴퓨팅, 클라우드 컴퓨팅, 모바일 서비스
E-mail: dykim723@gmail.com



라 현 정 (Hyun Jung La)
2003년 경희대학교 전자정보학부 졸업 (학사)
2006년 숭실대학교 컴퓨터학과 졸업 (석사)
2011년 숭실대학교 컴퓨터학과 졸업 (박사)
2011년 ~ 2013년 숭실대학교 모바일 서비스 소프트웨어공학센터 연구 교수
2013년 ~ 현재 ㈜스마티랩 대표
관심분야: 소프트웨어 아키텍처, 모바일 클라우드 컴퓨팅, 사물인터넷 컴퓨팅
E-mail: hjla80@gmail.com