韓國 컴퓨터情報學會 論文誌
第12卷 第1號, 2007. 3.

2007-12-1-1-5

모바일 에드혹 네트워크에서 안정성을 향상 시킨 분산 조합 가중치 클러스터링 알고리즘

황 윤 철*, 이 상 호**, 김 진 일***

Advanced Stability Distributed Weighted Clustering Algorithm in the MANET

Yoon-Cheol Hwang*, Sang-Ho Lee**, Jin-Il Kim***

요 약

모바일 에드혹 네트워크는 고정된 인프라의 도움 없이 이동 노드만으로 구성되므로 네트워크의 독립성과 융통성을 높일 수 있으나, 노드의 참여와 이탈의 자유로움 때문에 네트워크를 운영할 때, 망의 형태를 안정적으로 관리하는 것은 무엇보다도 중요하고 어려운 문제이다. 따라서 이러한 문제를 해결하기 위하여, 관리와 안전성에 중점을 둔 분산가중치 클러스터링 알고리즘을 제안한다. 제안된 알고리즘은 초기클러스터 형성시에는 기존의 분산가중치 알고리즘을 사용하고 클러스터 형성 후 이동 노드들로 인해 발생되는 재클러스터링을 최소한으로 줄이기 위해 부클러스터 헤드와 분산게이트웨이라는 개념을 사용한다. 성능 검증을 위해 초기의 오버헤드, 재가입률, 클러스터의 수를 기준으로 기존의 DCA과 WCA 알고리즘과 제안된 알고리즘을 비교, 평가한다.

Abstract

Mobile ad-hoc network(MANET) can increase independence and flexibility of network because it consists of mobile node without the aid of fixed infrastructure. But, Because of unrestriction for the participation and breakaway of node, it has the difficulty in management and stability which is a basic function of network operation. Therefore, to solve those problems, we suggest a distributed weighted clustering algorithm from a manageable and stable point of view. The suggested algorithm uses distributed weighted clustering algorithm when it initially forms the cluster and uses a concept which is distributed gateway and sub-cluster head to reduce the re-clustering to the minimum which occurs mobile nodes after forming the cluster. For performance evaluation, We compare DCA and WCA with the suggested algorithm on the basis of initial overhead, resubscriber rate and a number of cluster.

▸ Keyword : 모바일 에드혹 네트워크(MANET), 모바일(mobile), DCA, WCA, 클러스터링(clustering)

• 제1저자 : 황윤철, 교신저자 : 김진일(comclass@pcu.ac.kr)
• 접수일 : 2007.2.7, 심사일 : 2007.2.21, 심사완료일 : 2007. 3.13.
* 충북대학교 전자계산학과 박사과정 ** 충북대학교 전자계산공학과 교수
*** 배재대학교 교양교육지원센터 교수
※ 본 논문은 2006년 배재대학교 교내학술연구비로 수행된 것임.

Ⅰ. 서 론

모바일 에드혹 네트워크(MANET, Mobile Ad-hoc NETwork)는 기존의 무선 랜이나 이동통신망에서 사용되는 접근점(AP : Access Point)과 Base Station(BS)과 같은 기존 네트워크를 위한 기반 시설이 없이 네트워크에 참여하는 무선 전송 장비를 장착한 장비들 상호간의 통신에 의해 네트워크가 형성된다. 노드는 자유스럽게 이동하며 다수 홉으로 네트워크를 구성하고 라우터와 호스트 둘 다의 역할을 수행할 수 있다.

이러한 모바일 에드혹 네트워크의 망의 형태(Topology)는 노드들의 위치와 전송범위에 따라 자동적으로 구성되며 노드의 이동성과 무선 환경의 제한사항에 따라 동적인 특성을 갖고 있어서 망의 형태가 어떻게 형성되고 유지되는가에 따라 네트워크 성능에 많은 영향을 미친다. 그러므로 모바일 에드혹 네트워크 환경에서 네트워크 망의 형태를 가능한 안정적으로 관리하는 것은 무엇보다 중요하고도 어려운 문제이다[1][2]. 하지만, 모바일 에드 혹 네트워크가 가갖는 특성 즉, 무선 환경과 다중 홉을 통한 네트워크 구성, 제한된 배터리 운영, 그리고 노드의 이동성 등으로 인해 일반적인 네트워크 관리 구조로는 이러한 특성을 만족시킬 수 없다. 그러므로 라우팅 설정 시 오버헤드를 줄이고 라우팅 테이블의 크기를 줄일 수 있고 네트워크의 형태의 안정성을 확보할 수 있으며 매체 액세스 시 자원 관리나 대역폭 할당 등을 용이하게 하고 노드의 위치 관리나 송신 전력 관리를 가능하게 하는 클러스터 기반 구조가 타당한 접근 방법이다. 하지만, 기존에 나와 있는 노드의 식별자 (ID: identifier) 기반, 노드의 연결성(connectivity) 기반, 노드의 가중치(weight) 기반 클러스터링 알고리즘들은 클러스터가 형성된 후에도 클러스터 멤버 노드들의 이동으로 인해 클러스터의 망의 형태가 변경되거나 클러스터 헤드와의 연결이 끊길 경우 재 클러스터링을 수행하기 때문에 클러스터의 오버헤드가 증가하고 안정성이 떨어지는 문제가 발생한다.

따라서 본 논문에서는 클러스터를 형성할 때 초기 오버헤드를 줄이고 클러스터를 형성한 후에도 노드들의 이동으로 인해 발생하는 오버헤드를 줄이기 위해 분산조합가중치 클러스터링 알고리즘을 제안한다. 제안된 알고리즘은 클러스터 헤드를 선출하는데 있어 네트워크의 전체노드들에 대한 최소 가중치(Weight)를 계산하는 'global minima' 대신에 'local minima'를 사용하고 클러스터 형성 후에 발생되는 재가입율과 재클러스터링을 줄이기 위해 부클러스터 헤드와 분산 게이트웨이를 이용한다.

Ⅱ. 관련 연구

모바일 에드혹 네트워크에서는 자원의 효율적인 사용과 대역폭의 공간 재사용 제어, 다중 홉 네트워크에서 효과적인 서비스 품질 제공, 동적 네트워크 환경에서 제어를 목적으로 사용하는 정보(라우팅 테이블 등)를 유지하기 위해 필요한 데이터의 양을 최소화하려는 목적으로 클러스터링을 한다. 네트워크에서 클러스터링과 관련된 연구는 크게 클러스터 헤드의 선정을 통한 클러스터의 구성 알고리즘과 클러스터의 유지와 클러스터를 기반으로 하는 송신 전력 제어나 라우팅 기법들로 분류된다[1].

본 논문에서는 클러스터 헤드 선정에 관련된 연구를 중심으로 알아본다. 지금까지 제안된 클러스터 헤드 선정 알고리즘을 분류하면 세 가지 범주로 나눌 수 있는 데, 첫 번째 노드의 식별자 기반 클러스터 헤드 선정, 두 번째는 노드의 연결성 기반 클러스터 헤드 선정, 세 번째는 노드의 가중치 기반 클러스터 헤드 선정 알고리즘이다.

노드 식별자 기반의 클러스터 헤드 선정 알고리즘은 DARPA(Defense Advanced Research Project Agency)의 무선 패킷 라디오 네트워크에서 클러스터 구성을 하기 위해 제안한 연결 클러스터 알고리즘(LCA, Linked Cluster Algorithm)[3]으로 최고 식별자(highest ID) 기반과 최저 식별자(lowest ID) 기반 클러스터 헤드 선정으로 나눌 수 있다. 이 알고리즘은 이동성(mobility)를 고려한 네트워크 환경에서 네트워크 전체의 안정적 경로 유지를 위한 클러스터 기법으로 노드의 움직임, 노드나 링크의 장애, 노드의 추가에 의해 발생하는 연결도의 변화에 영향을 최소화하면서 중앙 제어 없이 분산처리를 통해 네트워크의 연결도를 주기적으로 감시하여 신뢰성 있는 네트워크 구조를 형성하고자 하였다. 최고 식별자 기반 클러스터링 알고리즘은 두 단계로 이루어진다. 첫 번째 단계에서 클러스터를 구성하고 두 번째 단계에서 클러스터 간에 링크를 생성한다. 알고리즘이 종료되면 노드들은 일반 노드, 헤드 노드, 게이트웨이 노드로 구분된다. M. Gerla[4]이 제안한 최저 식별자 기반의 클러스터 알고리즘은 클러스터 내에서 가장 작은 노드 번호를 가진 노드를 우선 클러스터헤드로 선정하여 클러스터를 형성함으로 알고리즘 간단하고 계산량이 적다는 장점을 가진 반면, 무조건 작은 노드 번호를 가진 노드가 클러스터 헤드가 됨으

로써 전체 노드간 균일한 부하분산을 이루지 못하고 클러스터헤드가 된 노드에게 많은 부하를 걸리게 함으로써 배터리의 소모를 증가 시켜 클러스터 헤드의 재선출 및 클러스터 재구성 가능성이 크다. Max-min d-cluster[5] 알고리즘은 노드 식별자를 기반으로 O(d) 시간 내에 d-홉 클러스터를 생성한다. d-홉 내에서 d라운드 동안 노드 식별자를 송수신하여 최대 식별자를 유지하고 다음 d라운드 동안 최소 식별자를 유지한다. 이후 저장된 식별자 중 여러 휴리스틱을 적용하여 클러스터 헤드를 결정한다. 이 알고리즘은 링크 클러스터 알고리즘에 비해 클러스터 헤드간의 부하 균형을 이루고 보다 적은 수의 클러스터를 생성한다. 또한 시간 동기를 필요로 하지 않는다.

최고 연결도(highest connectivity) 기반 클러스터링 알고리즘[6]은 각 노드에서 자신과 자신이 수신한 노드의 리스트를 브로드캐스트 하여 연결도가 가장 높은 노드를 클러스터 헤드로 선출하는 방식이다. 만약 연결도가 같은 경우라면 노드 식별자가 낮은 것을 우선하고,. 이미 클러스터 헤드로 결정된 노드는 더 이상 클러스터 구성에 포함시키지 않는다. 이 알고리즘은 같은 전송 범위에 있는 노드들 중 가장 연결성이 좋은 노드를 클러스터 헤드로 선정하므로 클러스터 헤드의 업데이트가 적은 반면에 하나의 클러스터 안에 노드들의 수가 제한되어 있지 않아 과부하가 발생할 수 있다. 또한, 각 클러스터는 각각의 구성원들 간에 공유되는 자원들을 균등하게 할당 받기 때문에 클러스터 안에 노드들이 늘어 날수록 클러스터 헤드가 처리해야하는 노드가 많아져 클러스터 헤드의 자원이 부족할 수도 있다.

노드의 가중치를 기반으로 클러스터를 구성하는 알고리즘에는 S. Basagni[7]가 제안한 분산 클러스터링 알고리즘(Distributed Clustering Algorithm, DCA)과 분산 이동성-적응적 클러스터링(Distributed Mobility-Adaptive Clustering, DMAC), Chatter과 Das, Turgut가 제안한 다중 변수를 기반으로 가중치를 결정하는 WCA(Weighted Clustering Algorithm)[8]이 있다.

가중치 기반 클러스터링 알고리즘은 기존 노드의 식별자나 연결도 기반 클러스터링이 클러스터링 하는 동안 노드의 이동성이 없다는 가정 하에 동작함으로써 실제 클러스터링 동안에도 끊임없이 노드의 이동성이 있는 애드 혹 네트워크를 수용하지 못한다고 반론하고 이를 위한 클러스터링 알고리즘으로 설계하였다. DCA는 이웃 노드 중 가중치가 가장 높은 노드가 클러스터 헤드로 결정되고 일반 노드는 클러스터 헤드 중 가중치가 가장 높은 클러스터 헤드에 결합(join)하게 된다. 노드가 자신이 헤드인지 일반노드인지를 결정하기 위해서는 자신보다 가중치가 높은 이웃노드의 결정이 이루어지기를 기다린다. 모든 노드는 구분되는 식별자를 가지고 있고 대응되는 가중치를 가진다. 이 알고리즘은 알고리즘이 실행되는 동안 네트워크의 망의 형태의 변화가 없다고 가정하기 때문에 노드의 이동성이 없거나 매우 느릴때 유용하다.

분산 이동성-적응적 클러스터링은 노드가 링크가 없어지거나 새로 발생하는 것을 감지할 수 있다고 가정하고 상태를 좀 더 세밀하게 구분하여 노드가 동작하도록 했다. 이 방법은 노드의 이동성이 충분히 고려되어 이동성에 반비례하게 가중치를 할당한 경우 노드의 식별자나 노드의 연결도 기반 클러스터링에 비해 클러스터 갱신이 적음을 알 수 있으나 노드의 가중치가 단계마다 변하기 때문에 클러스터헤드 선정 계산 시 오버헤드가 매우 높고 네트워크의 처리율이나 전력 제어에 최적화를 이룰 수 없다. 또한 클러스터 헤드 결정을 위해 이웃노드의 결정을 기다려야 하는 시간적 오버헤드가 존재한다.

위 두 방식은 최고 연결성 방식과 최저 노드번호 방식보다는 업데이트 횟수는 비교적 작으나 노드의 가중치가 계속 변화하기 때문에 주기적인 가중치 계산이 필요하고 노드 연결성이나 전력제어 같은 시스템 요소들에 대한 고려가 없는 것이 단점이다.

이후 M. Chatterjee et al.에 의해 다중 변수를 기반으로 가중치를 결정하는 WCA (Weighted Clustering Algorithm)가 제안되었다. 이 방식은 모든 노드들이 전체 네트워크에 있는 노드들의 가중치를 알아야 하기 때문에 클러스터를 형성하는데 시간이 길고 많은 오버헤드가 발생하고 생성된 클러스터헤드의 1-Hop 이웃 노드가 클러스터 헤드가 될 수 있어 클러스터들이 네트워크에 적절하게 분포되지 않는다.

이와 같은 단점을 해결하기 위해 K.Dhurandher, V. Singh에 의해 WBACA(Weighted Based Adaptive Clustering Algorithm)가 제안되었다. WCA 알고리즘과 마찬가지로 WBACA 알고리즘은 에드혹 특성에 따라 클러스터 헤드가 되기에 적합한 노드를 선정하기 위해 GPS(Global Positional System)을 이용한다. 이를 위해 전송 파워, 전송률, 이동성, 배터리 파워와 Degree를 계산하며 이를 매트릭으로 사용한다. WBACA 알고리즘은 WCA의 단점을 보안한 알고리즘으로써 클러스터 헤드들이 1-Hop 범위내의 이웃 노드가 되지 않도록 해서 클러스터들이 네트워크에 적절하게 분포되도록 하였다. 따라서 WCA보 다 클러스터링을 형성하는데 시간이 짧다. 그러나 클러스터 멤버 노드들의 이동으로 인해 클러스터의 망의 형태가

변경되거나 클러스터 헤드와의 연결이 끊길 경우 재 클러스터링을 수행하기 때문에 클러스터의 안정성이 떨어지는 단점이 있다.

대부분의 노드 가중치 기반 클러스터링 알고리즘들은 다른 클러스터링 알고리즘 보다 업데이트의 횟수가 적고 최적화된 방법으로 시스템 성능을 향상시키지만 업데이트가 발생하는 경우, 다른 클러스터링 알고리즘에 비해 계산량이 증가한다.

III. 안정성이 향상된 분산조합 가중치 클러스터링 알고리즘

본 논문에서 제안하는 분산 조합 가중치 클러스터링 알고리즘은 클러스터를 구성할 때, 분산화와 시스템의 유효시간 연장을 제공하는 데, 알고리즘의 동작은 초기화 과정을 거쳐 클러스터 형성 과정 그리고 클러스터 관리 과정으로 이루어진다.

3.1 기본 가정

먼저, 클러스터 헤더가 될 노드를 최적화하기 위해서는 다음 몇 가지 사항을 가정한다. 특히 알고리즘을 수행하는 동안에는 망의 형태 변화가 없다고 가정한다. 첫째, 각 클러스터 헤더는 효율적인 MAC(Medium Access Control) 기능을 위해 미리 지정된 시스템 한계인 N개의 노드를 이상적으로 지원할 수 있다. 둘째, 배터리 전력은 일반 노드로 사용될 때 보다 클러스터 헤더로 사용될 때 전력의 소모가 크다. 셋째, 일반 노드가 현재 있는 클러스터를 떠나서 다른 클러스터 지역으로 이동하면 재가입이 발생한다. 이 경우, 그 노드와 재가입 클러스터 사이의 정보 교환은 국부적으로 발생하며, 상대적으로 정보의 크기는 작다. 넷째, 클러스터 헤더는 주변의 노드가 전송 범위 이내에서 가까우면 가까울수록 더 나은 통신을 보장 받을 수 있으며, 거리가 멀어 질수록 감쇄한다. 다섯째, 클러스터 헤더는 상호 단일 홉 내에 존재할 수 없으며 두 홉 이상을 관리하며 클러스터 멤버는 클러스터 헤더로부터 항상 단일 홉 거리에 잇고 클러스터에 속한 일반 노드들은 서로 항상 최대 두 홉의 거리로 망의 형태를 구성한다. 여섯째, 클러스터 헤더로 선정된 특정 노드는 자신의 전원이 임계값에 도달하거나 갑자기전원이 커지지 않는 한 지속적으로 클러스터 헤더의 임무를 수행한다. 일곱 번째, 모든 노드는 HELLO 메시지를 주기적으로 이웃 노드로 전송하여 정보를 교환한다. 여덟 번째, 클러스터 멤버 노드는 자신의 클러스터 내에 잇는 모든 멤버들과 control packet(periodic notification packet)을 통해 정보를 교환한다. 아홉 번째, 클러스터의 게이트웨이 노드는 이웃 클러스터의 게이트웨이 노드와 클러스터 멤버의 수 등과 같은 정보 교환한다. 열 번째, RBDWCA에서는 시스템이 구성될 때 모든 노드는 자신만의 고유 노드번호(ID)와 (x,y)좌표로 구성되는 자신의 위치 정보를 가지고 있다.

3.2 초기화

시스템의 초기화 과정은 시스템이 활성화 될 때와 클러스터내의 각 노드의 전원이 소모되어 클러스터 재구성이 필요할 때 호출되며 다음과 같은 작업을 수행한다. 시스템이 초기화 되면 모든 노드들은 자신의 노드번호와 위치정보를 포함하는 HELLO 메시지를 전송하여 자신의 존재를 이웃 노드에게 알린다. 이웃 노드로부터 HELLO 메시지를 받으면, 해당 노드의 존재를 인식하게 된다. 모든 노드는 일정한 시간동안 HELLO 메시지를 받고 자신의 이웃 노드들의 집합 $N[v]$을 구성한다. 집합 $N[v]$은 이웃 노드들의 노드번호(ID)와 그에 상응하는 노드들의 위치좌표(x,y)로 구성되며 식 (1)과 같이 정의한다.

$$N[v] = k, P(xk, yk)) : k \in \text{이웃노드들의} ID \quad \cdots\cdots\cdots\cdots (1)$$

각 노드들은 자신의 가중치 W를 WCA에서 사용한 계산 단계와 동일하게 수행하고 나머지 항목만 클러스터 헤드로 동작하는 동안의 누적되는 시간 대신 실제로 노드에 남아있는 배터리 잔량을 사용함으로써 한정적인 자원인 배터리 전력을 보다 효율적으로 관리한다.

가중치 계산하는 과정을 단계별로 기술하면 다음과 같다.

1 단계 : 전송 범위(T_X) 내의 노드 연결성(Degree) 개수 dv를 식 (2)을 이용하여 계산한다.

$$dv = |N[v]| = \sum_{v' \in V, v' \notin v} \{dist(v, v') < T_X\} \quad \cdots\cdots\cdots\cdots (2)$$

그런 다음, 모드 노드 v에 대하여, 노드의 연결성 차를 식 (3)를 이용하여 계산한다. 클러스터 헤드가 처리할 수 있는 노드들의 수를 δ 라고 한다. 이것은 클러스터 헤드에 과부하가 걸리는 것을 방지하고, 시스템의 효율성을 관리하기 위해 설정한다.

$$\triangle v = |dv - \delta| \quad \cdots\cdots (3)$$

2 단계 : 모든 노드 v에 대하여 그 이웃 노드들과의 거리 합을 식 (4)에서 계산한다. Dv는 주로 에너지 소모와 관계가 있으며, 먼 거리에 있는 노드와 통신을 할 경우 원거리에 있는 노드와의 통신보다 더욱 많은 전력이 요구된다.

$$Dv = \sum_{v' \in N(v)} \{dist(v, v')\} \quad \cdots\cdots (4)$$

3 단계 : 현재 시간 T까지 모든 노드에 대하여 모든 v의 속도의 평균 Mv를 식 (5)을 이용하여 구한다. 시간 t에서 노드 v의 좌표는 (X_t, Y_t)이고 시간 (t-1)에서의 좌표는 (X_{t-1}, Y_{t-1})이다. Mv가 0이면 이동성이 전혀 없는 노드를 말한다.

$$Mv = \frac{1}{T}\sum_{t=1}^{T}\sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2} \quad \cdots\cdots (5)$$

4 단계 : 배터리의 잔량 Ev를 구한다. 이는 노드가 얼마나 많은 배터리 전력을 소모했는지 나타낸다.

5 단계 : 각 노드 v에 대하여 가중치 Wv를 식 (6)을 이용하여 구한다.

$$Wv = \omega_1 \triangle v + \omega_2 Dv + \omega_3 Mv + \omega_4 Ev \quad \cdots\cdots (6)$$

각 노드 v에 대하여 $Wv = \omega_1 \triangle v + \omega_2 Dv + \omega_3 Mv + \omega_4 Ev$이며 가장 작은 조합 가중치 Wv를 가진 노드가 클러스터 헤드가 되며, ω_1, w_2, w_3, w_4는 시스템 파라미터에 대응하는 가중치로 이들 계수의 합은 1이다.

각 노드들은 자신의 계산된 가중치 W를 포함하는 HELLO 메시지를 자신의 이웃 노드들에게 전송하고, 모든 노드는 일정 시간동안 HELLO 메시지를 받아 노드번호와 그에 상응하는 가중치로 구성된 집합 Z를 식 (7)과 같이 구성한다.

$$Z = \{(k, Wk) : k \in \{\text{이웃노드들의 } ID\} \cup \{\text{자신의 } ID\}\} \quad \cdots (7)$$

모든 노드들은 자신의 클러스터 헤드를 unknown으로 설정함으로써 초기화 과정을 마무리한다. 초기화 과정은 시스템이 활성화 될 때와 클러스터 헤드의 역할 포기에 의한 시스템 재구성시에만 수행된다.

3.3 클러스터 형성

클러스터링 형성 과정에서 발생하는 초기 설정 오버헤드를 줄이기 위해 제안한 알고리즘에서는 네트워크의 전체 노드들에 대한 최소 가중치를 계산하는 'global minima'를 사용하지 않고 'local minima'를 통해 클러스터 헤드를 선출하고 관리하는 방식을 사용한다. 다음은 클러스터 형성과정은 다음과 같다.

단계 1 : 모든 노드들은 HELLO 메시지를 브로드 캐스팅한다. 이때 HELLO 메시지는 해당 노드의 ID, 위치 정보, Wv를 포함한다.

단계 2 : 노드들은 자신의 모든 이웃 노드들로부터 HELLO 메시지를 받으면 자신의 조합 가중치인 Wv를 계산하며, 자신의 Wv를 변경한 후 수정된 HELLO 메시지를 브로드캐스트한다.

단계 3 : 자신의 1-Hop 범위내에 있는 이웃 노드들로부터 HELLO 메시지를 통해 Wv를 수신하면 자신의 Wv와 비교하여 자신의 역할을 결정한다.

1) 자신의 이웃 노드중 자기보다 작은 Wv가 없으면 자기 자신이 클러스터 헤드가 되고, HELLO 메시지의 정보 중 "$Node\_id = Ch\_id =$ 자기노드의 ID"로 변경해서 브로드캐스트한다.
2) 자신의 Wv 가 가장 작지 않다면 가장 작은 Wv를 가진 노드를 클러스터 헤드로 선정하고 이에 가입한다. 이때 메시지의 정보 중에서 "$Ch\_id =$ 해당클러스터헤드주소"로 변경하여 브로드캐스트 한다.
3) 주위에 클러스터 헤드가 없고 주위에 어떤 클러스터에도 가입하지 않은 노드들 중에서 자신의 Wv가 가장 작다면 자신이 클러스터 헤드가 된다.

단계 4 : 클러스터 헤드는 "$Ch\_id =$ 자기노드의 ID"인 HELLO 메시지를 받으면 해당 송신 노드를 자신의 클러스터 리스트에 추가한다.

단계 5 : 클러스터에 속한 노드들은 다른 클러스터 헤드의 선출과정에 참여하지 않는다.

단계 6 : 아직 클러스터에 가입하지 않은 나머지 노드들에 대해 단계 3에서 단계 5 과정을 반복하여 클러스터를 형성하고 모든 노드들이 클러스터에 속하게 되면 종료한다.

3.4 클러스터 관리

클러스터링이 완료되면, 클러스터 멤버 노드들은 주기적으로 클러스터 헤드에게 HELLO 메시지를 전송하여 망의 형태를 관리한다. 그러나 해당 클러스터에 속한 노드가 이동하거나 클러스터 헤드가 역할을 포기 했을 때는 망의 형태의 변화가 일어난다. 이런 경우에 변경된 클러스터에 대한 업데이트를 하기 위해 변경된 클러스터들에 대해 클러스터링 관리 과정이 수행된다. 먼저 해당 클러스터의 헤더가 과다한 배터리 낭비로 인해 자신의 역할을 포기할 때 클러스터를 관리하는 방법으로, 매번 클러스터 헤드가 자신의 역할을 포기 할 때마다 클러스터 헤드를 재선출 하게 되면 그에 따른 네트워크 프로세싱 오버헤드와 불안정한 망의 형태, 자원할당 및 데이터의 손실을 초래 할 수 있다. 따라서 본 논문에서는 클러스터 헤드가 선출된 다음에 클러스터 헤드가 자신의 단일 홉 내 이웃 노드 중 가중치 값이 자신보다 큰 노드들 중 가장 작은 노드를 부 클러스터 헤드로 선출하는 방법을 사용하며, 클러스터 헤드에 일정한 한계치를 적용하여 그 한계치 이상일때만 클러스터 헤드의 역할을 수행하게 하고 한계치에 도달하면 부 클러스터 헤드가 클러스터 헤드의 임무를 대행하게 함으로써 클러스터 헤드를 재선출 과정에서 발생하는 네트워크 프로세싱 오버헤드 및 선출 지연, 자원 할당 및 데이터의 손실을 최소화 한다. 다음으로 클러스터 내의 해당 노드들의 이동으로 인해 클러스터 재구성 문제가 발생했을때 클러스터 관리 방법으로 매번 클러스터내의 해당 노드들 중의 일부가 해당 클러스터 외로 이동할 때마다 클러스터를 재구성하면 네트워크에 잦은 오버헤드와 정보전송의 지연이 발생하게 된다. 따라서 본 논문에서는 분산게이트웨이라는 역할을 이탈하고 있는 노드와 가장 가까운 클러스터내의 노드에게 부여하여 클러스터내의 해당 노드가 클러스터 외로 이동 할 때 그 이동 노드를 관리하게 함으로써 클러스터 재구성이 발생하지 않도록 한다. 앞에서 나열한 방법에 대하여 클러스터 헤드가 역할을 포기했을 경우에 클러스터를 관리하는 방법에 대해 먼저 설명하고, 그런 다음에 클러스터내의 해당 노드 중에서 일부가 이동했을 때의 클러스터 관리방법에 대해 설명한다.

3.4.1 클러스터헤드의 역할 포기시 클러스터링관리

모든 노드에 대하여 클러스터가 형성되면, 각각의 클러스터 내의 클러스터 헤드는 자신의 단일 홉 이웃 노드 중 가중치가 가장 작은 노드를 부 클러스터 헤더로 선정한다. 클러스터 헤드는 초기에 망의 형태를 구성할 때, 이웃 노드와의 가중치를 비교하여 선출되며, 사전에 정의된 임계값 범위 내에서 클러스터 헤드로서의 역할을 수행한다. 각 노드들이 클러스터 헤드로의 역할 분담을 위해 미리 설정되는 임계값(Threshold : 이하 Th)은 식 (8)과 같다.

$$Th = \alpha \times ACW \quad (\alpha \le 1.0) \cdots\cdots (8)$$

여기서, α 는 LBF(Load Balancing Factor), ACW는 부 클러스터 헤드의 가중치를 의미한다.

그림 1에서 보는 것처럼, 클러스터 헤드인 노드 1의 가중치가 한계값 이하로 내려가거나 전원이 갑자기 꺼진 경우 부 클러스터 헤드인 노드 2가 클러스터 헤드가 되어 클러스터를 재구성하는 데, 전의 클러스터 헤드 노드인 1을 제외한 나머지 노드 중 가중치가 가장 작은 노드를 부 클러스터 헤드로 선정한다. 클러스터 헤드 v_i가 부 클러스터 헤드 AC를 선출하는 방법은 식 (9)와 같다.

$$AC = \min\{weight(Z_i - v_i)\} \cdots\cdots (9)$$

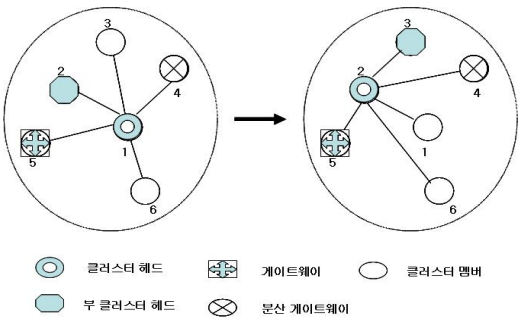

그림 1. 클러스터 헤드의 변경
Fig. 1. The change of Cluster head

클러스터 헤드가 된 노드 2는 클러스터 ID를 자신의 ID로 변경한 후 자신이 속한 클러스터 노드들에게 브로드캐스트 하고 이를 수신한 노드들은 자신의 정보를 갱신하고 자신의 상태를 변경한다. 이러한 과정을 도식화하면 그림 2와 같다.

3.4.2 클러스터구성 노드 이동시 클러스터링 관리

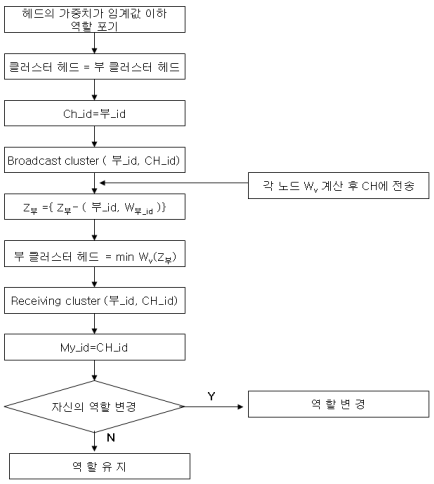

그림 2. 클러스터헤드의 역할 포기
Fig. 2. The coversion of cluster head

클러스터를 구성하는 노드들이 이동하였을 경우에 망의 형태가 변경되고 그에 따라 변경된 클러스터에 대한 업그레이드를 위해 클러스터링 관리 과정이 수행된다. 기존 알고리즘들[8][9][10][11]의 경우에는 클러스터가 형성된 후 클러스터를 구성하는 노드가 이동해서 클러스터 헤드로부터 연결이 끊길 경우 클러스터를 재구성하는 과정을 수행했다. 이에 따라 클러스터를 구성하는 노드들이 이동할 때마다 클러스터 헤드와 연결이 끊기게 된다면 클러스터의 변경이 발생하여 불안정하게 된다. 본 논문에서 제안하는 알고리즘에서는 클러스터 구성 노드가 이동시 클러스터 헤드와 연결이 끊기더라도 자신과 같은 클러스터에 속해 있는 해당 노드와 가장 가까이 있는 노드가 분산게이트웨이가 있는 경우에 원래 클러스터 헤드와 계속해서 연결 되도록 하여 노드 이동시 클러스터의 변경이 최소화 되도록 하였다. 즉 클러스터헤드와 직접 연결이 끊긴 노드라 할지라도 자신과 같은 클러스터에 속해 있는 분산게이트웨이와 연결되어 있다면 분산게이트를 통해 자신의 클러스터헤드와 계속해서 연결이 되도록 한다.

노드 번호가 Wv라고 가정하고 분산게이트웨이를 사용하지 않은 경우와 사용한 경우의 클러스터 변화에 대해 알아본다. 먼저 그림 3처럼 분산게이트웨이를 사용하지 않는 경우, 클러스터 멤버인 노드 4가 이동함으로써 클러스터 헤드인 노드 1과 연결이 끊기게 되고 노드 4는 클러스터 형성 과정을 가지게 된다. 자신의 1-hop 범위에 있는 인근 노드들 중 Wv가 가장 작은 노드3을 클러스터 헤드로 선출하게 되고 노드4는 클러스터 헤드가 3인 노드의 클러스터 멤버가 된다. 노드 6도 현재 클러스터 헤드인 노드 5보다 노드 3의 Wv가 작기 때문에 클러스터 헤드가 3인 노드의 클러스터 멤버가 된다. 그런 후 나머지 노드들에 대해 클러스터링이 재구성 된다. 따라서 노드의 이동은 자신의 클러스터 구성 뿐만 아니라 다른 클러스터의 구성에도 영향을 끼치게 되어 전반적으로 클러스터의 안전성을 떨어뜨린다.

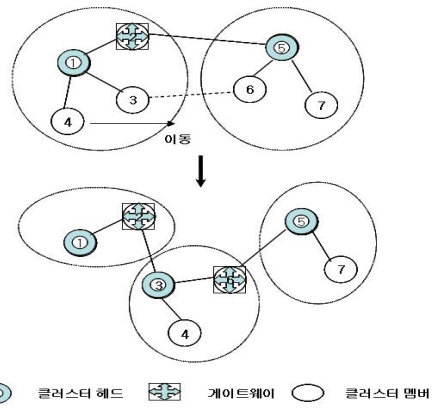

그림 3. 분산게이트웨이를 적용하지 않은 경우
Fig. 3. In Inapplicable case of distributed gateway

반면에, 그림 4처럼 분산게이트웨이를 사용할 경우 노드 4가 이동하여 노드 1과 연결이 끊긴다고 하더라도 노드 3을 통해 계속해서 자신이 속해 있는 클러스터에 속하게 됨으로써 클러스터 변경이 생기지 않는다. 따라서 노드의 이동으로 인하여 자신의 속해있는 클러스터에 역할 분담을 초래할 뿐 다른 클러스터에게는 영향을 끼치지 않게 됨으로 전체 네트워크의 안전성을 높인다.

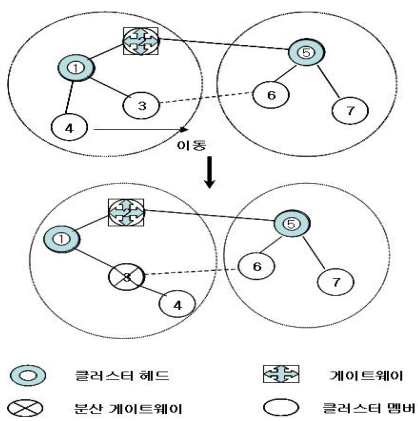

그림 4. 분산 게이트웨이를 적용하는 경우
Fig. 4. In applicable case of distributed gateway

그림 5는 클러스터를 구성하는 노드들이 이동하였을 경우 본 논문에서 제안하는 클러스터 과정을 도식화 한 것이다.

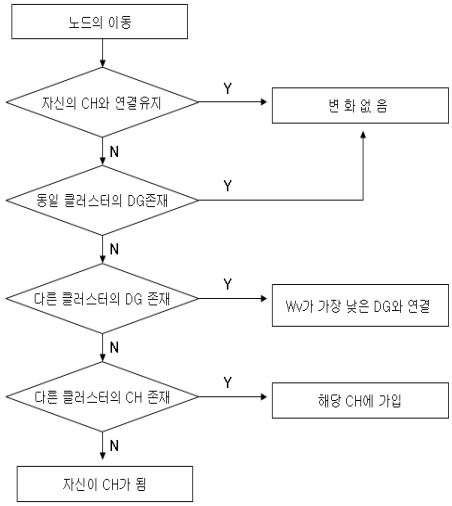

그림 5. 노드 이동시 클러스터링 관리 흐름도
Fig. 5. Clustering management flowchart

Ⅳ. 실험결과 및 고찰

성능평가를 위한 시뮬레이션은 UCLA 대학에서 유무선 네트워크를 시뮬레이션 하기 위해 개발한 GloMoSim (Global Mobile Information System Simulator) V2.03을 통해 구현하였다. GloMoSim은 여러 개의 라이브러리 모듈로 구성되어 있으며 이들 각 모듈은 특정 프로토콜을 시뮬레이션 하는데 사용한다. GloMoSim의 라이브러리는 UCLA 대학에서 개발한 시뮬레이션 언어인 PARSEC[13]으로 개발 되었으며 새로운 모듈과 프로토콜들은 프로그램을 수정하여 사용한다.

4.1 실험 환경

본 논문에서 제안한 알고리즘의 시뮬레이션 환경은 표 1과 같다.

표 1. 시뮬레이션 환경 변수
Table 1. Simulation parameters

환 경 변 수	값
네트워크 크기	1000 * 1000m
노드의 개수	30,60.90.120,150개
노드의 배치	임의로 선택
노드의 이동성	10ms
노드의 전송 반경	250m
대 역 폭	2Mbps
노드의 이동속도	uniform하게 최소 0m/s에서 최대 10m/s
Radio Propagation model	Free space
MAC protocol	802.11
Network protocol	AODV
CBR	2 패킷/초, 512 bytes
조합 가중치 요소	w_1=0.7, w_2=0.2, w_3=0.05, w_4=0.05
연결성 임계치(δ)	3

4.2 평가 결과

노드의 이동속도를 10m/s 라고 가정하고 클러스터의 수, 초기 구성 시 발생하는 오버헤드, 재가입자수를 이용하여 기존의 DCA, WCA 알고리즘과 비교 평가한다. 클러스터 수는 전체 노드 수에 따라 형성되는 평균 클러스터 수를 표시하는 것으로 노드들의 이동속도와 밀집도에 많은 영향을 받는다. 동일한 이동 속도와 동일한 밀집도를 사용한다고 가정하고, 노드의 수가 60일 때의 결과를 보면 그림 6에서 보는 것처럼, 제안한 알고리즘이 기존의 알고리즘보다 약 8%정도의 클러스터를 적게 생산함을 볼 수 있다.

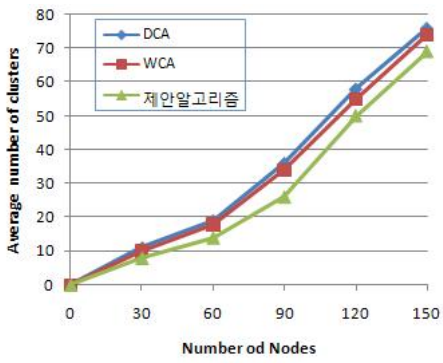

그림 6. 평균 클러스터의 수
Fig. 6. Average number of clusters

클러스터를 구성하기 위해 초기에 발생하는 오버헤드 측면을 보면, 그림 7에서 보는 바와 같이, 기존 알고리즘들은 노드가 증가 할수록 초기에 드는 오버헤드 비용이 증가하는 반면 제안한 알고리즘은 노드가 증가하는 것과는 상관없이 일정한 값을 유지하는 것을 볼 수 있다. 그 이유

는 제안한 알고리즘은 전송 범위 밖의 정보는 다 버리고 클러스터를 형성하는 이웃 정보만을 사용하기 때문이다.

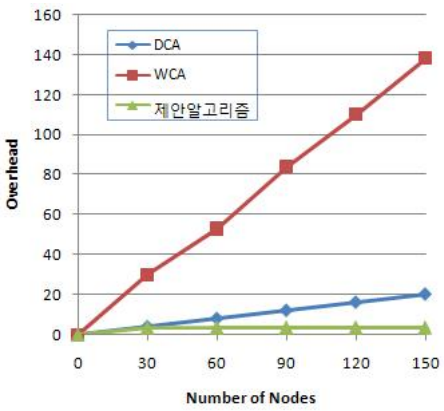

그림 7. 초기 클러스터의 오버헤드
Fig. 7. Overhead of initial cluster

재결합수(reaffiliations)는 노드의 이동성을 인해 발생되며 노드의 수가 작았을 경우는 선형적으로 증가하다가 노드의 수가 증가하면 재결합수가 천천히 줄여 드는 것을 볼 수 있다. 노드의 개수가 30일 때는 WCA 보다 27% 정도의 재결합률이 낮고, 노드의 수가 120일 때는 18% 정도의 재결합률이 낮게 나타났다. 그 원인은 제안한 알고리즘에서는 클러스터의 안정성을 되도록 오래 유지할 수 있도록 부 클러스터 헤드와 분산 게이트웨이를 사용했기 때문이다.

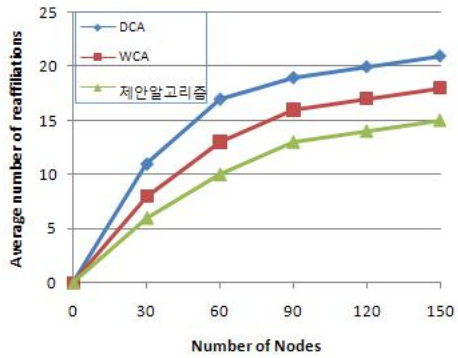

그림 8. 평균 재결합수
Fig. 8. Average number of reaffiliations

V. 결 론

모바일 에드 혹 네트워크은 기지국과 같은 고정 기반망의 도움 없이 여러 이동 단말기 간의 통신을 통해서 이루어지는 단일 혹은 다중 홉의 임시적인 네트워크이다. 이동 단말기들이 자신의 전송 범위 안에 있는 주변 단말기에게 데이터를 전달해가면서 네트워크 전체에 걸친 라우팅이 가능하도록 데이터 전송이 이루어진 모바일 에드 혹 네트워크는 주로 기반 망이 설치되기 어려운 환경에서 사용되는 임시망의 성격을 갖는다. 특히 무선 인터페이스의 사용과 동적 망의 형태, 중앙 통제 기관의 부재로 인해 많은 문제가 발생되는데 이를 효율적으로 다룰 수 있는 구조가 절실히 요구된다. 따라서 본 논문에서는 모바일 에드 혹 네트워크에서 발생되는 여러 가지 문제를 효율적으로 해결하기 위한 방법으로 계층적 구조를 선택했으며 이동 노드들 간의 계층 구조를 형성하기 위해 필요한 분산 조합 클러스터링 알고리즘을 제안했다. 이 알고리즘은 기존의 알고리즘과는 달리 클러스터를 구성하는데 있어 인접 1홉 범위내의 노드들만으로 클러스터를 구성함으로써 초기 클러스터 구성시 발생하는 오버헤드를 감소하고, 구성된 클러스터를 가능한 안정성을 오래 지속할 수 있도록 부 클러스터 헤드와 분산 게이트웨이라는 개념을 도입하여 재클러스터링의 횟수를 줄였다. 그 결과 클러스터의 개수와 초기 클러스터에서 발생하는 오버헤드, 그리고 클러스터 생성 후 발생되는 재결합수가 기존의 알고리즘 보다 적음을 시뮬레이션을 통해 증명하였다.

향후 연구과제로는 제안된 알고리즘은 단지 제한된 조건내에서 시뮬레이션을 한 결과로 차후에 실제 환경에서 얼마만큼의 안정성을 발휘하는지에 대한 연구가 필요하다.

참고문헌

〔1〕 Abhay K. Parekh, "Selecting routers in ad-hoc wireless networks", IEEE, 1995
〔2〕 Lichun Bao, "Topology Management in Ad Hoc
〔3〕 D. J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a

Distributed Algorithm," Comm. IEEE, Vol. 29, No. 11, pp. 1694-1701, 1981.
〔4〕 Gerla. M, and Tsai, J. T. C,"Multicluster, mobile, multimedia radio network", ACM/Baltzer Journal of Wireless Networks 1, 3, pp.225-265, 1995.
〔5〕 A. D. Amis, R. Prakash, Thai H. P. Vuong, and D. T. Huynh, "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks," Proc. of IEEE INFOCOM, Vol. 1, pp.32-41, 2000.
〔6〕 Chien-Chung Shen, "A Cluster-Based Topology Control Framework for Ad Hoc Networks", IEEE, 2004.
〔7〕 S. Basagni, "Distributed Clustering for Ad Hoc Networks," Proc. of International Symposium on Parallel Architectures, Algorithms and Networks, pp. 310-315, 1999.
〔8〕 Mainak Chatterjee, Sajal K. DAS, "WCA: A Weighted Clustering Algorithm for mobile Ad Hoc Networks", Cluster Computing, Vol.5 No.2, pp.193-204, 2002
〔9〕 S. Corson, J. Macker, ""Mobile Ad-hoc Networking (MANET),"", Internet Draft, IETF, October 1998.
〔10〕 S. K. Dhurandher, G. V. Singh, "Weight Based Adaptive Clustering in Wireless Ad Hoc Networks," Personal Wireless Communications, 2005. ICPWC 2005. 2005 IEEE International Conference on 23-25, January 2005, pp. 95 --100.
〔11〕 Wonchang choi and Miae Woo,"A Distributed weighted Clustering Algorithm for mobile Ad Hoc Networks",AICT, In proceedings of IEEE Computer Society, 2006.2

저 자 소 개

황 윤 철

1994년: 한남대학교 전자계산공학과 졸업(공학사)
1996년: 한남대학교 대학원 전자계산공학과졸업(MS)
1999년~현재: 충북대학교 대학원 전자계산학과 박사 과정수료
〈관심분야〉 네트워크, IDS, ITS, Ad Hoc 및 센서네트워크 보안

이 상 호

숭실대학교 대학원 전자계산학과 졸업 (PHD)
1976년 1월~1979년 5월: 한국 전력 전자계산소
1981년 6월~현재: 충북대학교 전기전자 및 컴퓨터공학부 교수
〈관심분야〉 Protocol Engineering, Network Security, Network Management, Network Architecture

김 진 일

한남대학교 컴퓨터공학과 공학박사
배재대학교 IT센터 책임강사
(주)블루베리소프트 개발팀장
현, 배재대 교양교육지원센터 교수
〈관심분야〉 병렬처리, 네트워크, 퍼지이론, 교육 콘텐츠