이동 애드혹 네트워크에서 로드 밸런싱을 위한 클러스터링 기법

임 원 택[†] · 김 구 수[††] · 김 문 정[†††] · 엄 영 익[††††]

요 약

이동 애드혹 네트워크는 이동 호스트들만으로 이루어진 자율적인 네트워크로 유비쿼터스 컴퓨팅의 연구와 더불어 활발하게 연구되고 있다. 이동 애드혹 네트워크의 평면적인 구조를 개선하기 위해, 계층적으로 네트워크를 관리하는 클러스터링 기법들이 연구되고 있다. 하지만 기존 기법들은 멀티홉 클러스터링과 로드밸런싱에 관한 연구가 미약하다. 본 논문에서는 다중 홉 클러스터링을 지원하고, 클러스터 헤드 간의 로드 밸런싱을 고려한 클러스터링 기법을 제안한다. 이를 위해 이동 애드혹 네트워크에서 나타날 수 있는 클러스터의 분열 현상과 이에 따른 클러스터의 상태를 정의하고, 로드 밸런싱을 위한 참가, 병합, 분열, 클러스터 헤드 선출 등의 기법을 제안한다.

키워드 : 애드혹 네트워크, 클러스터링, 로드 밸런싱

The Clustering Scheme for Load-Balancing in Mobile Ad-hoc Network

Won Taek Lim[†] · Gu Su Kim[††] · Moon Jeong Kim[†††] · Young Ik Eom[††††]

ABSTRACT

Mobile Ad-hoc Network(MANET) is an autonomous network consisted of mobile hosts. A considerable number of studies have been conducted on the MANET with studies of ubiquitous computing. Several studies have been made on the clustering schemes which manage network hierarchically to improve flat architecture of MANET. But the conventional schemes have the lack of multi-hop clustering and load balancing. This paper proposes a clustering scheme to support multi-hop clustering and to consider load balancing between cluster heads. We define the split of clusters and states of cluster, and propose join, merge, divide, and election of cluster head schemes for load balancing of between cluster heads

Key Words : Ad-hoc Network, Clustering, Load Balancing

1. 서 론

이동 애드혹 네트워크는 기존에 존재하는 유선 네트워크나 기간망을 사용하지 않고, 이동 호스트만으로 이루어진 자율적인 네트워크이다[1]. 이동 애드혹 네트워크는 기존의 기간망을 사용하는 네트워크와는 달리 필요시에 빠르게 구성될 수 있는 특징을 가진다. 이러한 특징 때문에 유선 네트워크를 구성하기 힘든 군사 지역에서의 통신이나, 인명 구조와 같은 긴급 상황에서의 통신 등의 분야에 이동 애드혹 네트워크가 활용될 수 있다[2].

하지만 이동 애드혹 네트워크는 기간망을 사용하는 기존의 네트워크에 비해 많은 문제점을 가지고 있다. 이동 애드혹 네트워크의 가장 큰 문제점 중의 하나는 네트워크 정보를 유지하기 위한 컨트롤 패킷에 의한 오버헤드가 크다는 것이다. 이동 애드혹 네트워크는 호스트들 간의 계층이 없는 평면적 구조를 가지므로 다른 노드와 통신을 하기 위해서는 주기적으로 자신의 정보를 브로드캐스팅 하거나 다른 노드를 찾기 위한 패킷을 전체 네트워크에 플러딩해야 하는 오버헤드가 발생한다[3].

이러한 문제점을 개선하기 위해 이동 애드혹 네트워크의 클러스터링 기법이 제안되었다. 클러스터링 기법이란 전체 이동 애드혹 네트워크를 가상의 그룹인 클러스터로 나누어 네트워크를 계층적으로 관리하는 기법이다. 클러스터로 나누어진 계층적인 구조는 이동 애드혹 네트워크를 보다 효율적으로 관리할 수 있도록 해 준다[4].

지금까지 제안된 대부분의 클러스터링 기법들은 클러스터를 관리하는 클러스터 헤드와 클러스터 멤버가 서로 인접해 있는 단일 홉 클러스터링 기법이다[5-9]. 이러한 단일 홉 클러스터링 기법은 잦은 클러스터 헤드의 교체를 필요로 하며,

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 IT연구센터 지원사업의 연구결과로 수행되었음(IITA-2005-(C1090-0501-0019))
† 준 회 원 : 성균관대학교 대학원 정보통신공학부 석사과정
†† 준 회 원 : 성균관대학교 대학원 정보통신공학부 연구원
††† 준 회 원 : 고려대학교 정보보호기술센터(CIST) 정보보호대학원 연구원
†††† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2006년 7월 2일, 심사완료 : 2006년 8월 5일

이에 따라 클러스터 유지 오버헤드가 증가한다. 또한 한 클러스터 헤드에 지나치게 많은 노드가 연결될 수 있는 문제점을 가지고 있다[10].

본 논문에서는 다중 홉 클러스터링을 지원하고, 클러스터 헤드 간의 로드 밸런싱을 고려한 클러스터링 기법을 제안한다. 이를 위해 이동 애드혹 네트워크에서 나타날 수 있는 클러스터의 여러 가지 상태를 정의하고, 이를 기반으로 클러스터의 상태를 정상적인 상태로 유지하기 위한 병합, 분할, 참가, 클러스터 헤드 선출 등의 작업들을 제안하였다. 이와 같은 작업을 통해 이동 애드혹 네트워크에 존재하는 클러스터는 일정한 크기와 정상적인 상태를 유지할 수 있다.

본 논문의 2장에서는 기존에 연구된 로드 밸런싱에 관련된 클러스터링 기법을 설명한다. 3장에서 이동 애드혹 네트워크를 구성하는 구성 요소들의 구조를 살펴보고 4장에서는 로드 밸런싱을 위해 클러스터 헤드가 수행하는 작업들과 이에 따른 통신 복잡도에 대해 살펴본다. 5장에서는 제안 기법에 대한 시뮬레이션 결과를 분석하고 6장에서 결론을 맺는다.

2. 로드밸런싱을 고려한 클러스터링 기법

2.1 DLBC (Degree-Load-Balancing Clustering)

DLBC[9]는 각 클러스터에 존재하는 이동 노드의 수를 일정하게 유지하기 위해서 주기적으로 클러스터링을 수행하는 기법이다. 이를 위해 클러스터 헤드가 조종할 수 있는 최적의 수인 Elected Degree라는 값과 이동 노드 수의 범위를 제한하는 값인 Max-Delta 값을 정의한다. 클러스터에 존재하는 이동 노드의 수는 언제나 Elected Degree ± Max-Delta값 이내에 있어야 하며 만일 이 범위를 넘어서게 되면 해당 클러스터의 클러스터 헤드는 클러스터 헤드의 자격을 상실한다.

DLBC는 로드 밸런싱이 적용되지 않은 일반적인 LCA(Linked Cluster Algorithm)[12] 보다 클러스터 헤드의 지속성이 약 25% 증가하였다. 이는 클러스터 헤드가 일정한 차수의 이상의 노드만을 가지고 있으면 클러스터 헤드가 변경될 필요가 없기 때문이다. 하지만 클러스터 헤드가 바뀌게 되면 주변의 클러스터에도 연속적인 영향을 미치는 Ripple effect 현상이 발생하는 단점을 가진다[3].

2.2 AMC (Adaptive Multi-hop Clustering)

AMC[10]는 멀티홉 클러스터를 지원하는 로드 밸런싱 클러스터링 기법이다. AMC에서 각 노드는 주기적으로 자신의 상태와 노드 ID, 클러스터 ID 정보를 교환한다. 노드들은 크게 클러스터 멤버, 클러스터 헤드, 게이트웨이 노드로 나누어진다. 클러스터 헤드는 클러스터 멤버의 수를 조절한다. 게이트웨이 노드는 인접한 다른 클러스터의 게이트웨이 노드와 클러스터 정보를 교환하고, 클러스터 헤드에게 이를 보고한다. 이와 같은 과정을 통해 노드들은 클러스터의 토폴로지 정보를 알 수 있다.

AMC는 클러스터 크기의 경계를 지정하고 클러스터의 크기가 경계를 벗어나면 병합과 분할 작업을 수행한다. 이와 같은 과정은 클러스터의 크기를 일정한 범위 내에서 유지시켜 준다. 하지만 클러스터 분할 시에 클러스터 헤드를 선출하는 명확한 알고리즘이 설명되어 있지 않고, 노드 이동성에 의한 클러스터의 분열에 관해서는 고려하지 않는다.

3. 네트워크 구성

본 장에서는 애드혹 네트워크를 나타낼 수 있는 표현을 정의하고, 애드혹 네트워크를 구성하는 클러스터와 노드의 특성과 구조에 대해 설명한다.

3.1 애드혹 네트워크

이동 애드혹 네트워크를 구성하는 요소들은 방향성 없는 그래프(undirected graph) G = (V, E)로 나타낼 수 있다. 그래프에서 노드(vertex) V는 이동 애드혹 네트워크를 구성하는 노드들의 집합이다. $v_i \in V$ 인 v_i는 노드의 ID가 i인 이동 노드를 나타낸다. E는 이동 애드혹 네트워크를 구성하는 노드들 간을 연결하는 무선 링크들의 집합을 나타낸다. $(v_i, v_j) \in E$ $(i \neq j)$는 노드의 ID가 i인 노드와 j인 서로 다른 노드 간을 연결하는 무선 링크를 나타낸다.

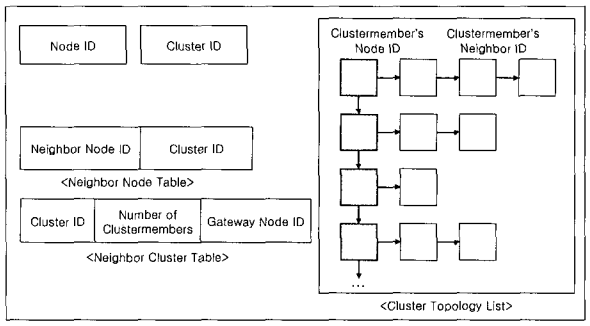
3.2 클러스터의 구조

본 논문에서 제안하는 클러스터링 기법은 디스조인트 멀티-홉 클러스터링 기법이다. 따라서 이동 애드혹 네트워크를 구성하는 노드는 하나 이상의 클러스터에 속할 수 없고, 클러스터의 크기는 2홉 이상이 될 수 있다. 본 기법은 IP 계층에서 제공되는 라우팅 기법과 독립적으로 운용되며, 애드혹 네트워크를 구성하는 노드들은 클러스터 정보를 교환하기 위해 각 노드의 정보와 클러스터링 정보를 주기적으로 주고받는다.

클러스터 C_h는 $C_h \subset V(v_h \in V)$와 같이 나타낼 수 있다. C_h는 클러스터 헤드가 v_h인 클러스터를 나타낸다. 클러스터의 크기는 클러스터를 구성하고 있는 노드의 개수로, $Size(C_h)$로 나타내기로 한다. 정상적인 상태에서는 클러스터 헤드 v_i는 클러스터 C_i에 속하지만, 노드의 이동에 따라 클러스터가 둘로 나누어지면 클러스터 헤드 v_i가 클러스터 C_i에 속하지 않는 경우가 생길 수도 있다. 클러스터의 크기는 최대 임계치(T_{high})와 최소 임계치(T_{low}) 사이에서 유지된다. 만일 클러스터의 크기가 최대 임계치보다 커지면 분할 작업을, 최소 임계치보다 작아지면 병합 작업을 수행한다. 클러스터 상태의 변화에 관한 내용은 4장에서 자세히 다룬다.

3.3 노드의 구조와 상태

이동 애드혹 네트워크를 구성하는 노드들은 클러스터를 유지하기 위해서 많은 종류의 정보를 유지한다. 각각의 노드가 유지하는 정보를 (그림 1)에서 보인다.



(그림 1) 각 노드가 유지하는 정보

노드 ID와 클러스터 ID는 하나의 변수로써 유지되는 정보이다.

- **Node ID** : 노드들을 구별하기 위한 식별자이다. 노드 ID는 노드가 생성될 때 함께 생성되며 한 이동 애드혹 네트워크에서 유일하다. 노드 v의 노드 ID는 NodeID(v)로 나타내기로 한다.
- **Cluster ID** : 노드가 속한 클러스터를 나타내는 식별자이다. 클러스터 ID는 클러스터 헤드의 노드 ID와 같다. 노드 v의 클러스터 ID는 Cluster(v)로 나타내기로 한다.

애드혹 네트워크와 클러스터의 상태를 알기 위해 두 개의 테이블과 하나의 인접 리스트(adjacency list)를 유지한다.

- **Neighbor Node Table** : 이웃 노드의 노드 ID와 클러스터 ID를 저장하는 테이블이다. Neighbor Node Table의 항목은 이웃 노드로부터 주기적으로 발생되는 신호를 감지함으로써 추가되고, 일정한 시간 동안 신호가 감지되지 않으면 삭제된다. Neighbor Node Table의 정보는 클러스터 토폴로지 정보를 구성하고, 게이트웨이 노드의 역할을 수행하는데 사용된다.
- **Neighbor Cluster Table** : 인접한 클러스터의 정보를 저장하는 테이블로, 클러스터 헤드만이 이 정보를 유지한다. Neighbor Cluster Table의 항목은 이웃 클러스터 헤드로부터 주기적으로 전송되는 패킷을 수신함으로써 추가되고, 일정한 시간 동안 패킷이 전송되지 않으면 삭제된다. Neighbor Cluster Table의 정보는 클러스터의 병합 시에 사용된다.
- **Cluster Topology List** : 클러스터의 토폴로지를 나타내는 정보이다. Cluster Topology List는 클러스터 멤버의 노드 ID와 해당 클러스터 멤버의 이웃 노드들의 노드 ID로 구성된다. Cluster Topology List의 항목은 각 노드가 주기적으로 발생하는 노드의 정보를 담은 패킷을 수신함으로써 갱신되며 저장된 정보는 클러스터의 분열의 감지와 클러스터 헤드 선출 시에 사용된다.

애드혹 네트워크를 구성하는 노드는 클러스터 구성에 따라 여러 가지 상태를 가질 수 있다. 노드 v_i의 상태는 State(v_i)로 나타기로 한다. 노드가 가질 수 있는 상태는 다음과 같이 정의된다.

- **CM(Cluster Member)** : Cluster(v_i) ≠ Cluster(v_j)이고 (v_i, v_j) ∉ E 일 때 Cluster(v_i) ≠ NodeID(v_i) 이면 State(v_i) = CM이다. State(v_i) = CM인 노드를 클러스터 멤버라고 한다. 이 상태는 클러스터 헤드도, 게이트웨이 노드도 아닌 상태로, 클러스터 멤버는 자신이 가진 노드 정보만을 브로드캐스팅 한다.
- **GW(Gateway)** : Cluster(v_i) ≠ Cluster(v_j)이고 (v_i, v_j) ∈ E 일 때 Cluster(v_i) ≠ NodeID(v_i) 이면 State(v_i) = GW이다. State(v_i) = GW인 노드를 게이트웨이 노드라고 한다. 게이트웨이 노드는 인접한 클러스터의 정보를 자신이 속한 클러스터의 클러스터 헤드에게 전송하는 역할을 한다.
- **CH(Cluster Head)** : Cluster(v_i) ≠ Cluster(v_j)이고 (v_i, v_j) ∉ E 일 때 Cluster(v_i) = NodeID(v_i) 이면 State(v_i) = CH 이다. State(v_i) = CH인 노드를 클러스터 헤드라고 한다. 한 노드가 클러스터 헤드로 선출되거나 클러스터 크기가 1일 때 클러스터 헤드가 된다. 클러스터 헤드는 클러스터 정보를 인접 클러스터로 주기적으로 전송하고 클러스터의 병합 및 분할 등 클러스터의 재구성 여부를 판단하는 역할을 한다.
- **CHGW(Cluter Head & GateWay)** : Cluster(v_i) ≠ Cluster(v_j)이고 (v_i, v_j) ∈ E 일 때 Cluster(v_i) = NodeID(v_i) 이면 State(v_i) = CHGW 이다. State(v_i) = CHGW인 노드는 클러스터 헤드와 게이트웨이 노드가 수행해야 하는 작업을 모두 수행한다. 하지만 게이트웨이 노드가 클러스터 헤드로 전송하는 정보는 목적지가 자기 자신이기 때문에 전송하지 않는다.

4. 로드 밸런싱을 위한 클러스터 관리 기법

클러스터는 클러스터의 크기와 클러스터 헤드의 유무에 따라 여러 가지 상태를 가질 수 있다. 클러스터 헤드는 클러스터의 상태를 정상적인 상태로 유지하기 위해 병합, 분할, 클러스터 헤드 선출과 같은 여러 가지 이벤트를 발생시킨다. 본 장에서는 클러스터가 가질 수 있는 상태를 정의하고, 클러스터 상태를 변화시킬 수 있는 이벤트들에 대하여 설명한다.

4.1 클러스터 상태 및 상태 변화

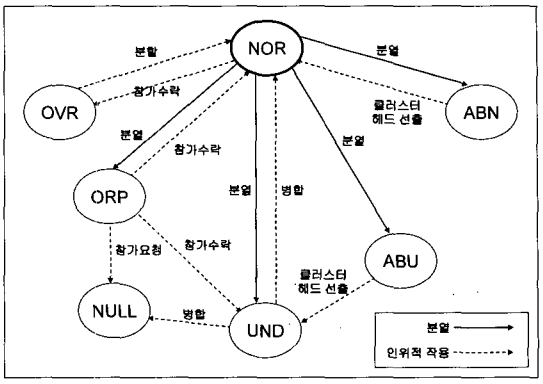
클러스터가 분열되면 분열된 클러스터는 각각 다른 상태를 가지게 된다. 이러한 상태는 클러스터의 크기와 클러스터 헤드의 포함 여부에 따라 결정된다. 클러스터가 가질 수 있는 상태를 <표 1>에서 보인다.

〈표 1〉 클러스터의 상태 정의 및 설명

상태	정의	설명
NOR (Normal)	($T_{low} \leq Size(C_i) \leq T_{high}$) ∧ ($v_i \in C_i$)	클러스터의 크기가 임계치 내에 있고 클러스터 헤드를 포함하는 클러스터의 상태로, 정상적인 클러스터 상태이다.
ORP (Orphan)	$Size(C_i) = 1$	클러스터 멤버가 하나인 클러스터의 상태이다. 네트워크 구성 초기 노드들이나 노드의 이동에 의해 클러스터로부터 이탈한 하나의 노드가 가지는 상태이다.
UND (Under)	($1 < Size(C_i) < T_{low}$) ∧ ($v_i \in C_i$)	두 개 이상의 클러스터 멤버로 구성된 클러스터의 크기가 최소 임계치 미만이면서 클러스터 헤드를 포함한 경우이다. 클러스터 헤드는 주변 클러스터와의 병합을 시도하게 된다.
OVR (Over)	($Size(C_i) > T_{high}$) ∧ ($v_i \in C_i$)	클러스터의 크기가 최대 임계치를 초과하면서 클러스터 헤드를 포함하는 경우이다. 클러스터 헤드는 클러스터를 두 개로 나누는 클러스터 분할을 수행한다.
ABN (Abnormal)	($T_{low} \leq Size(C_i) \leq T_{high}$) ∧ ($v_i \notin C_i$)	클러스터의 크기가 임계치 내에 있지만 클러스터 헤드를 포함하지 않는 클러스터의 상태이다. ABN 상태를 감지한 노드에 의한 클러스터 헤드 선출을 통해 NOR 상태로 변하게 된다.
ABU (Abnormal-under)	($1 < Size(C_i) \leq T_{low}$) ∧ ($v_i \notin C_i$)	두 개 이상의 클러스터 멤버로 구성된 클러스터의 크기가 최소 임계치 미만이면서 클러스터 헤드를 포함하지 않는 경우이다. ABU 상태를 감지한 노드에 의한 클러스터 헤드 선출을 통해 UND 상태로 변하게 된다.
NULL	-	참가나 병합에 의해 클러스터가 사라진 상태이다. ORP상태였던 클러스터가 다른 클러스터에 참가하게 되거나, 두 클러스터가 병합 되었을 경우, 두 개의 클러스터가 하나의 클러스터로 합쳐지기 때문에 두 클러스터 중 하나의 클러스터는 사라지게 된다. 이러한 상태를 표현하기 위한 상태가 NULL 상태이다.

클러스터 헤드는 클러스터의 상태를 NOR 상태로 유지하려는 성질을 가지고 있다. 클러스터의 상태는 클러스터의 분열에 의해 계속해서 바뀌게 되는데, 클러스터의 상태가 바뀔 때 마다 클러스터 헤드는 클러스터의 상태를 계속적으로 점검한다. 만일 클러스터의 상태가 NOR상태가 아니라면 클러스터 헤드는 클러스터의 상태를 NOR상태로 바꾸기 위해 참가, 병합, 분할, 헤더 선출 중 적절한 작업을 수행한다. 이와 같은 상태 변화는 상태 전이도로 나타낼 수 있다. 클러스터의 상태 전이도를 (그림 2)에서 보인다.

ORP, UND, ABU, ABN 상태는 모두 분열로 인해 NOR 상태로부터 해당 상태로 전이된다. OVR 상태는 외부로부터 ORP 상태의 노드가 참가하게 되면 NOR 상태로 전이된다. NOR상태의 클러스터가 ORV상태가 됨에도 불구하고 ORP 상태의 클러스터의 참가요청을 받아들이는 이유는 로드 밸런싱 때문이다. OVR 상태에 가까워진 클러스터는 임계치에 가까운 노드를 관리해야 하기 때문에 로드 밸런싱이 필요한



(그림 2) 클러스터의 상태 전이도

상태이다. 이 때 ORP인 노드의 참가를 수락하고 분열 작업을 수행하게 되면, 분열된 두 개의 클러스터가 노드를 나누어 관리하기 때문에 로드 밸런싱이 가능해 진다.

ORP 상태의 클러스터나 UND 상태의 클러스터는 다른 클러스터에 흡수되어 NULL 상태가 될 수 있다. 참가를 요청하는 클러스터와 참가를 수락하는 클러스터 모두 ORP 상태일 경우, 참가를 수락하는 ORP 상태의 클러스터는 최소 임계치에 따라 NOR 상태나 UND 상태로 전이한다.

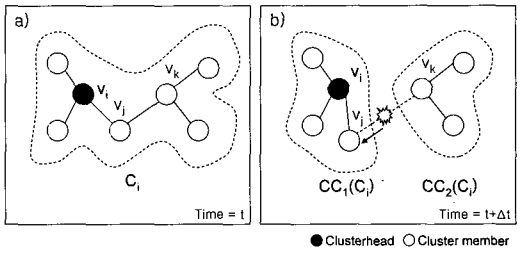
4.2 클러스터의 분열

클러스터의 분열은 클러스터 헤드가 발생시키는 이벤트가 아니라 이동 애드혹 네트워크를 구성하는 노드의 이동성 때문에 발생하는 이벤트이다. 클러스터 분열을 정의하기 위해 몇 가지 표현을 정의한다.

- **SubGraph(C_i)** : $V' = \{v_i \mid v_i \in C_i\}$ 이고 $E' = \{(v_i, v_j) \mid v_i \in C_i \text{ and } v_j \in C_i\}$ 인 V'와 E'로 이루어진 그래프를 $SubGraph(C_i) = (E', V')$ 라고 한다.
- **$CN(C_i)$** : $SubGraph(C_i)$의 connected component의 수
- **$CC_n(C_i)$** : $SubGraph(C_i)$의 n번째 connected component, $CN(C_i) = n$ 일 때 $CC_1(C_i)$, ... $CC_n(C_i)$ 가 존재

위에서 정의한 표현들을 이용하여 클러스터의 분열을 정의하면 다음과 같다.

- **클러스터 C_i 가 임의의 시간 t와 t+Δt 사이에 분열되었다 ⇔ t일 때 $CN(C_i) = 1$ 이고 t+Δt 일 때 $CN(C_i) > 1$ 이다.**



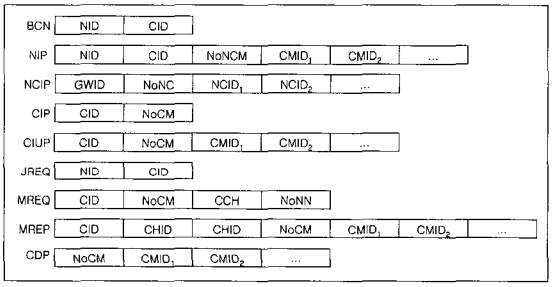
(그림 3) 클러스터 분열의 예

클러스터 분열의 예를 (그림 3)에서 보인다. (그림 3)의 a)는 시간 t일 때 분열되지 않은 클러스터의 예를 보여준다. 클러스터 C_i를 구성하는 노드들은 모두 무선 링크를 통해 통신이 가능하므로 $CN(C_i) = 1$이다. (그림 3)의 b)는 Δt가 흐른 후 분열된 클러스터의 모습을 보여준다. Δt가 흐른 동안 v_j가 이동하여 v_k와의 무선 링크가 끊어졌다. 이 때 클러스터 C_i의 SubGraph(C_i)는 두 개의 connected component를 가지므로 $CN(C_i) = 2$ 이다.

클러스터의 분열이 발생하면 분열된 각각의 클러스터는 자신의 상태에 따라 적절한 작업을 수행한다. 예를 들어, (그림 3)의 b)에서 $CC_1(C_i)$의 경우, 클러스터의 크기가 7에서 4로 작아졌으므로 사전에 설정된 임계치에 따라 클러스터를 병합해야 할지에 대한 여부를 결정해야 한다. $CC_2(C_i)$의 경우, $CC_2(C_i)$를 구성하는 노드들의 클러스터 ID 값은 i 이지만 v_i와 통신이 불가능하므로 클러스터 헤드를 선출하는 작업이 필요하다.

4.3 패킷의 종류와 구조

클러스터 구조를 유지하기 위해서 여러 종류의 패킷이 사용된다. 본 제안 기법에서 사용되는 패킷의 구조를 (그림 4)에서 보인다.



(그림 4) 클러스터 정보를 유지하기 위해 사용되는 패킷

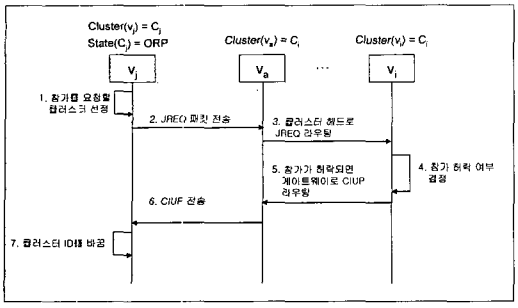
- **BCN(Beacon)** : 인접한 클러스터에게 자신의 존재를 알리는 패킷으로자신의 Node ID(NID)와 Cluster ID(CID)로 구성된다.
- **NIP(Node Information Packet)** : 클러스터 멤버에게 인접한 클러스터 멤버의 리스트를 전달하는 패킷이다. NIP는 자신의 Node ID (NID), Cluster ID (CID), 인접한 클러스터 멤버의 수 (NoNCM), 인접한 클러스터 멤버의 ID들($CMID_n$)로 구성되며, 클러스터 멤버 노드들에게만 전달된다.
- **NCIP (Neighbor Cluster Information Packet)** : 게이트웨이 노드가 자신의 클러스터 헤드로 인접 클러스터의 ID를 전달하는 패킷이다. NCIP는 게이트웨이의 Node ID (GWID), 인접한 클러스터의 개수 (NoCN), 인접 클러스터의 ID들($NCID_n$)로 구성된다.
- **CIP (Cluster Information Packet)** : 클러스터 헤드가 인접 클러스터 헤드로 자신의 클러스터 멤버의 수를 전달하는 패킷이다. CIP는 자신의 Cluster ID(CID), 자신의 클러스터 멤버의 수(NoCM)로 구성된다.
- **CIUP (Cluster Information Update Packet)** : 클러스터를 새로 구성해야 할 때, 새로 구성된 클러스터 멤버에게 전달되는 패킷이다. CIUP는 새로 구성된 클러스터의 Cluster ID(CID), 새로 구성된 클러스터의 클러스터 멤버의 수(NoCM), 새로 구성된 클러스터의 클러스터 멤버 ID ($CMID_n$)로 구성된다.
- **JREQ (Join Request)** : 클러스터의 상태가 ORP인 클러스터가 인접 클러스터로 참가를 요청하는 패킷이다. JREQ는 요청하는 노트의 Node ID(NID), 참가를 바라는 클러스터의 Cluster ID(CID)로 구성되며, 대상 클러스터의 클러스터 헤드로 전달된다.
- **MREQ (Merge Request)** : 클러스터의 상태가 UND인 클러스터의 클러스터 헤드가 인접 클러스터와 병합을 요청하는 패킷이다. MREQ는 병합을 요청하는 클러스터의 Cluster ID(CID)와 클러스터 멤버의 수(NoCM), 후보 클러스터 헤드(CCH) 그리고 후보 클러스터 헤드의 인접 노드의 개수 (NoNN)로 구성되며, 요청 대상 클러스터 헤드로 전송된다.
- **MREP (Merge Reply)** : MREQ를 수신한 클러스터 헤드가 병합을 요청한 클러스터 헤드로 전송하는 패킷이다. MREP는 자신의 Cluster ID(CID), 새로 선출된 클러스터 헤드의 ID (CHID), 자신의 클러스터 멤버의 수 (NoCM), 자신의 클러스터 멤버의 ID들 (CMIDn)로 구성된다. 만일 클러스터 병합 요청을 거부할 경우 자신의 Clsuter ID(CID)와 병합 요청 거부의 이유를 담아 전송한다.
- **CDP (Cluster Divide Packet)** : 클러스터의 상태가 OVR일 경우 한 클러스터를 두 개의 클러스터로 나누기 위해 사용되는 패킷이다. CDP는 새로운 클러스터 헤드로 전송되며 새로 구성된 클러스터의 클러스터 멤버의 개수 (NoCM), 새로 구성된 클러스터의 클러스터 멤버 ID들(CMIDn)로 구성된다.

4.4 클러스터 상태 전이 이벤트

분열을 제외한 모든 이벤트들은 모두 클러스터 헤드나 특정 노드가 특정한 작업을 수행해 주어야하는 이벤트이다. 이와 같은 이벤트의 종류에는 참가, 분할, 병합, 클러스터 헤드 선출이 있다.

4.4.1 참가

참가는 하나의 노드로 이루어진 클러스터가 다른 클러스터로의 참가를 요청하는 과정이다. (그림 5)는 클러스터의 참가 과정의 시퀀스 다이어그램을 보인다. 참가를 원하는 클러스터 헤드는 Neighbor Cluster Table을 참조하여 참가를 요청할 클러스터를 선정한다. 선정되는 클러스터는 인접 클러스터들 중 클러스터 크기가 가장 작은 클러스터이다. 이와 같은 기준으로 클러스터를 선정하는 것은 클러스터의 로드 밸런싱을 가능하게 해 주며, 클러스터의 크기가 분열로 인해 계속해서 작아지는 것을 막아준다.



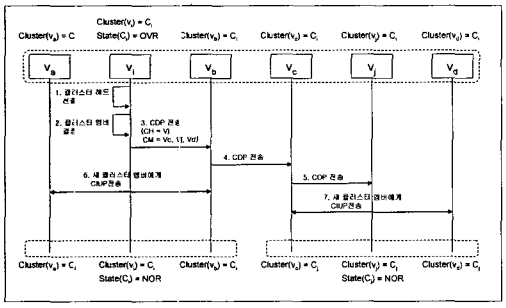
(그림 5) 클러스터 참가 시의 시퀀스 다이어그램

JREQ를 받은 클러스터 헤드는 클러스터로의 참가 허락 여부를 결정한다. 만일 참가를 허락한다면 JREQ를 보낸 노드를 클러스터 멤버에 합류시키고 클러스터 멤버 모두에게 CIUP를 전송한다. 참가를 허락하지 않을 경우는 참가 요청을 무시하고 아무 작업도 수행하지 않는다. 참가를 요청한 노드는 JREQ를 보내고 일정 시간이 지나도 CIUP를 받지 못하면 다른 클러스터에게 참가를 요청한다.

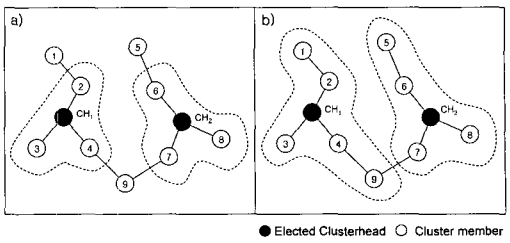
4.4.2 분할

분할은 클러스터의 Size(C_i) > T_{high} 일 때 클러스터 C_i의 클러스터 헤드인 노드 v_i가 자신의 클러스터를 적절한 크기로 나누는 과정이다. (그림 6)은 클러스터 분할 과정의 시퀀스 다이어그램을 보인다. 클러스터 C_i의 클러스터 헤드 노드인 노드 v_i는 클러스터의 상태가 OVR이 된 것을 감지하고 분할 과정을 시작한다. 분할 이후 클러스터의 클러스터 헤드와 클러스터 멤버를 선정하고 선정된 정보를 CDP에 실어 새로 선출된 클러스터 헤드에게 전송한다. CDP를 전송받은 새 클러스터 헤드는 클러스터 멤버에게 CIUP를 전송함으로써 새로운 클러스터가 생성되었음을 클러스터 멤버에게 알린다. CIUP를 전송받은 클러스터 멤버는 자신의 클러스터 ID와 클러스터 멤버 정보를 갱신한다.

클러스터의 분할을 시작하는 클러스터 헤드는 두 후보 클러스터 헤드와 각 클러스터 헤드가 가지게 될 클러스터 멤버를 선정한다. 구체적인 클러스터 헤드 선출 과정은 4.4.4절에서 설명하기로 한다. 분할 될 클러스터의 클러스터 멤버를 설정하는 것은 선출된 두 클러스터 헤드로부터 1홉씩 클러스터의 범위를 확장해 나감으로써 이루어진다. (그림 7)은 클러스터 멤버의 선정 과정의 예를 보인다.



(그림 6) 클러스터 분할 시의 시퀀스 다이어그램



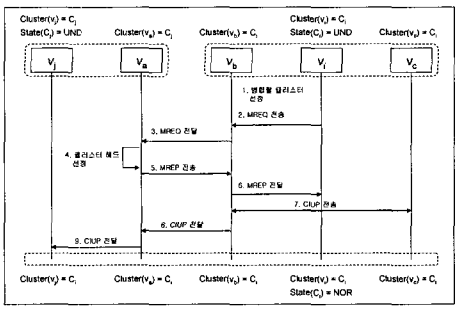
(그림 7) 클러스터 멤버 선정의 예

(그림 7)의 모든 노드는 모두 하나의 클러스터에 속한 노드들이고, 이 중 검은색으로 칠해진 CH_1, CH_2는 클러스터 헤드 선출 알고리즘에 의해 선출된 노드들이다. 선출된 각각의 클러스터 헤드의 클러스터 멤버는 (그림 7)의 a)에서와 같이 1홉 범위에서 (그림 7)의 b)에서와 같이 2홉 범위로 클러스터 영역을 확장해 나감으로써 결정된다. 2홉 범위에서 모든 클러스터 멤버가 결정되었으므로 클러스터 멤버 설정 과정은 종료된다. (그림 7)에서 9번 노드는 어느 CH_1이나 CH_2에 모두 속할 수 있지만 CH_1이 먼저 클러스터의 범위를 결정하였기 때문에 CH_1에 속하게 된다.

4.4.3 병합

병합은 클러스터의 상태가 UND인 두 클러스터가 하나의 클러스터로 합쳐지는 과정이다. (그림 8)은 클러스터의 병합 과정의 시퀀스 다이어그램을 보인다. 클러스터 병합을 요청하는 클러스터 헤드는 이웃 클러스터들 중 병합을 요청할 클러스터를 선정하여야 한다. 참가의 경우와 마찬가지로 병합을 요청할 클러스터도 Neighbor Cluster Table을 참조하여 인접 클러스터들 중 클러스터의 크기가 가장 작은 클러스터를 선택한다. 병합할 클러스터가 선택되면 자신의 클러스터에서 후보 클러스터 헤드를 선출하고, 선출된 후보 클러스터 헤드의 ID를 MREQ에 실어 병합할 클러스터의 클러스터 헤드로 전송한다.

MREQ를 전송받은 클러스터 헤드는 병합을 받아들일지에 대한 판단을 하고 그 결과를 MREP에 실어 전송한다. 만일 병합을 받아들일 경우 자신의 클러스터에서 후보 클러스터 헤드를 선출해 MREQ를 통해 전송받은 후보 클러스터 헤드와 비교해 가장 적합한 클러스터 헤드를 선출한다. 이렇게 선출된 클러스터 헤드의 정보는 MREP를 통해 상대 클러스터 헤드로 전송된다. 만일 병합 후의 클러스터 멤버



(그림 8) 클러스터 병합 시의 시퀀스 다이어그램

가 임계치를 넘어가거나, ABN, ABU의 상태에 있다면 클러스터의 병합을 거절한다. 병합의 거절 이유는 MREP 패킷에 실려 전송된다.

MREP를 수신한 클러스터 헤드는 선정된 클러스터 헤드를 새 클러스터 헤드로 하는 CIUP패킷을 새 클러스터 멤버에게 전송함으로써 병합 과정을 마친다.

4.4.4 클러스터 헤드 선출

분열된 클러스터는 반드시 클러스터 헤드가 존재하지 않은 클러스터를 만들어낸다. 4.1절에서 설명한 바와 같이 클러스터 헤드가 존재하지 않는 클러스터는 비정상적인 상태로 간주된다. 이러한 클러스터에서는 반드시 클러스터 헤드가 선출되어야 한다. 이 밖에도 클러스터 헤드 선출 알고리즘은 클러스터의 병합이나 분열 과정에서도 사용된다. (그림 9)에서 클러스터 헤드 선출 알고리즘을 보인다.

```
NodeID ClusterHeadElection(graph Subgraph)
// Elect new Cluster Head and return new Cluster Head ID
maxDgree = 0
newClusterHeaderID = ∞
for each node n in Subgraph do
  if degree of n >= maxDegree then
    if degree of n == maxDegree and n > newClusterHeadID then
      continue
    endif
    maxDegree = degree of n
    newClusterHeaderID = n
  end if
end for
return newClusterheaderID
```

(그림 9) 클러스터 헤드 선출 알고리즘

제안 기법에서는 클러스터 C의 클러스터 헤드의 선출 기준은 SubGraph(C)에서의 각 노드의 이웃 클러스터 멤버의 개수와 NodeID이다. 우선 클러스터 C의 노드로 이루어진 그래프인 SubGraph(C)를 구한다. 그리고 SubGraph(C)를 구성하는 각 노드의 이웃 클러스터 멤버의 개수를 비교하여 이 값이 가장 큰 노드를 클러스터 헤드로 선출한다. 이와 같은 기준을 적용하는 이유는 클러스터 헤드의 이탈을 방지하여 분열의 횟수를 최대한 줄이기 위해서이다. 만일 같은 수의 이웃 노드를 가진 노드가 존재한다면 NodeID가 작은 노드가 우선순위를 갖는다.

4.5 제안 기법의 통신 복잡도

제안 기법의 통신 복잡도는 단위 시간 Δt 동안 발생하는 평균적인 패킷의 수에 대한 복잡도를 구한 것이다. 이를 구하기 위해 몇 가지 값을 정의하였다. N은 네트워크에 존재하는 전체 노드의 개수를 나타내며 n은 평균 클러스터 멤버의 수를 나타낸다. m은 클러스터의 개수를, gw는 클러스터에 존재하는 게이트웨이 노드의 수를 나타낸다. e는 단위 시간 Δt 동안 발생되는 클러스터 헤드변경 이벤트의 평균 회수이다. 이 때 BCN, NIP, NCIP는 Δt 간격으로 브로드캐스팅 된다고 가정한다.

Δt 동안 발생되는 BCN 패킷과 NIP 패킷은 네트워크에 존재하는 모든 노드로 브로드캐스팅 되므로 각각 N개의 패킷이 발생한다. NCIP는 각 게이트웨이에서 클러스터 헤드로 유니캐스팅 되므로 gw개의 패킷이 발생하고, CIP는 한 클러스터 헤드로부터 이웃 클러스터 헤드로 전송되므로 모든 클러스터가 연결된 최악의 경우 m(m-1)개의 패킷이 발생한다. CIUP는 이벤트가 발생했을 경우 클러스터 멤버들에게 브로드캐스팅 되므로 e번의 빈도로 n개의 패킷을 발생시킨다. 이 외의 패킷은 모두 이벤트가 발생하는 기간 동안 유니캐스팅 되므로 CIUP의 복잡도에 상수로 포함된다. 이와 같은 결과를 모두 종합하게 되면 클러스터를 유지하기 위한 통신 복잡도는 다음과 같다.

$$O(2N+m^2+gw+e \cdot n)$$

5. 성능평가

본 장에서는 제안 기법의 성능평가의 결과에 대해 설명한다. 제안 기법의 성능평가는 시뮬레이션을 통해 이루어졌으며, 노드의 이동 속도에 관계없이 일정한 클러스터 크기를 가지는 안정성, 노드의 이동 속도와 클러스터 크기의 임계치에 따른 클러스터 유지 오버헤드, 그리고 본 제안 기법의 효율성을 알아보기 위해 여러 가지 성능 지표들을 측정하였다.

5.1 시뮬레이션 환경

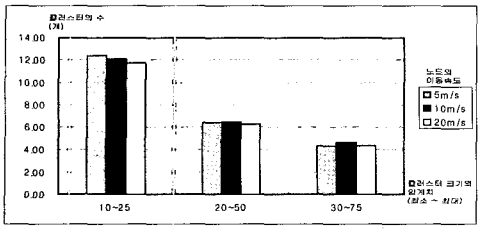
시뮬레이션은 가로 750m, 세로 750m의 정방형 공간에서 200개의 노드들이 존재하는 환경에서 이루어졌다. 각 이동 노드들은 실험에서 요구하는 이동성의 정도에 따라서 각각 5m/s, 10m/s, 20m/s의 속도로 이동한다. 각 시뮬레이션은 모두 60초 동안 네트워크에 존재하는 클러스터의 개수의 평균, 각 클러스터 크기의 평균, 클러스터 헤드의 변화 회수, 병합과 분할이 이루어진 회수를 측정하였다.

시뮬레이션을 시작하면 초기의 모든 노드들은 ORP의 상태로 서로에게 참가와 병합을 요청하며 클러스터를 구성한다. 이때의 네트워크는 참가 작업과 병합작업이 급격하게 이루어지는 불안정한 상태로, 이때 발생한 참가와 병합 작업은 클러스터를 유지하기 위한 작업이 아니라 클러스터를 구성하기 위한 작업이다. 본 시뮬레이션은 클러스터를 유지하는데 필요한 작업의 회수나 클러스터의 상태를 측정하기 위한 것이므로, 시뮬레이션 초기의 1초의 불안정한 네트워크 상태에서 발생한 데이터들은 모두 무시하고 1초 이후의 네트워크 상태만을 측정하였다.

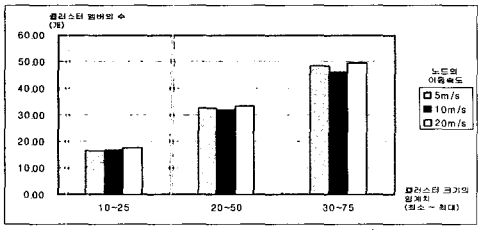
5.2 클러스터 크기의 안정성

노드의 이동 속도에 관계없이 클러스터 크기의 임계치에 따라 클러스터의 크기가 얼마나 일정하게 유지되는지를 측정하기 위해 0.5초마다 네트워크에 존재하는 클러스터의 개수와 각 클러스터의 클러스터 멤버 수를 측정하여 평균을 구하였다. 시뮬레이션은 노드의 이동 속도가 각각 5m/s, 10m/s, 20m/s일 때 클러스터 크기의 임계치가 10~20, 20~50, 30~75인 상황에서 진행되었다. (그림 10)과 (그림 11)은 클러스

터 크기의 안정성에 대한 시뮬레이션의 결과를 보인다.



(그림 10) 클러스터 크기 임계치에 따른 평균 클러스터 수



(그림 11) 클러스터 크기 임계치에 따른 평균 클러스터 멤버의 수

클러스터의 개수는 클러스터 크기의 임계치가 커짐에 따라 작아지는 경향을 보였다. 클러스터 크기의 임계치가 10~20일 때 평균 12.07, 20~50일 때 평균 6.34, 30~75일 때 평균 4.45개로 나타났다. 이동성에 따른 클러스터 수의 차이는 최대 평균대비 4.1%로 대체적으로 노드의 이동 속도에 관계없이 클러스터의 크기가 일정하게 유지되었다.

클러스터 멤버의 수는 클러스터 크기의 임계치가 커짐에 따라 증가하는 경향을 보였다. 클러스터 멤버의 수는 클러스터 크기의 임계치가 10~20일 때 평균 16.88, 20~50일 때 평균 32.64, 30~75일 때 평균 47.89로 나타났다. 이동성에 따른 클러스터 멤버의 수의 차이는 최대 평균 대비 3.9%로, 클러스터의 개수와 마찬가지로 노드의 이동 속도에 관계없이 일정하게 유지되었다.

5.3 클러스터 유지 오버헤드

클러스터를 유지하기 위한 오버헤드를 측정하기 위해서 클러스터 헤드의 변화 횟수와 병합과 분할 작업이 이루어진 횟수의 평균값을 측정하였다. 시뮬레이션은 노드의 이동속도가 20m/s인 상태에서 300초 동안 진행되었고, 시작 직후 5초 동안의 평균값은 나타내지 않았다. (그림 12)와 (그림 13)은 클러스터 유지의 오버헤드에 관한 시뮬레이션 결과를 보인다.

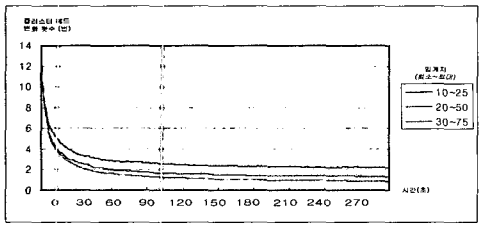
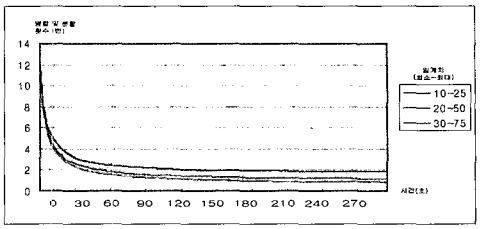


그림 12 시간에 따른 평균 클러스터 헤드 변화 횟수



(그림 13) 시간에 따른 평균 병합 및 분할 횟수

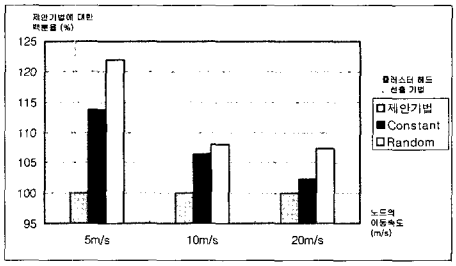
클러스터 헤드의 변화 횟수와 병합 및 분할 작업의 횟수는 시간이 지남에 따라 일정한 값으로 수렴하는 것을 볼 수 있다. 이와 같은 결과는 초기에 클러스터 구성이 끝나고 나면 클러스터를 유지하기 위한 오버헤드는 일정하게 유지된다는 사실을 알려준다. 클러스터 유지 오버헤드는 클러스터 크기가 클수록 작아지는 경향을 나타냈는데, 그 이유는 클러스터의 크기가 커질수록 다른 클러스터와의 통신 경로가 많아져, 클러스터의 분열 현상이 클러스터의 크기가 작을 경우보다 상대적으로 적게 일어나기 때문이다.

또한, 클러스터를 유지하는데 필요한 오버헤드는 클러스터 크기 임계치가 커질수록 감소하였다. 이는 클러스터의 크기가 커짐에 따라 안정적인 클러스터를 구성할 수 있음을 의미한다. 이와 같은 결과는 1홉 이상의 클러스터의 크기를 지원하는 멀티홉 클러스터링의 장점을 잘 나타낸다.

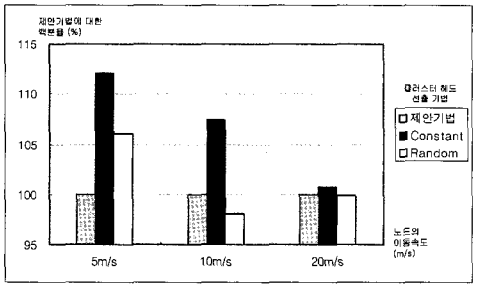
5.4 제안 기법의 효율성

제안 기법의 효율성을 측정하기 위해 두 가지의 다른 클러스터 헤드 선출방식과 제안 기법을 비교하였다. 비교할 클러스터 헤드 선출 방식은 클러스터가 병합작업을 수행할 때 언제나 병합을 요청받는 클러스터의 클러스터 헤드로 선출하는 방식과, 클러스터의 병합작업이나 분할작업, 클러스터의 분열 시에 선출되는 클러스터 헤드를 임의적으로 설정하는 방식이다.

시뮬레이션은 클러스터 크기의 임계치가 30~75일 때, 각 기법에서의 클러스터 헤드의 변화 횟수와 클러스터의 병합 및 분할 횟수를 측정하였다. 이를 기반으로 제안 기법의 오버헤드를 100으로 놓았을 때의 각 기법의 클러스터 헤드의 변화 횟수와 병합과 분할 작업 횟수를 백분율로 계산하였다. (그림 14)와 (그림 15)는 제안 기법의 효율성을 측정하기 위한 시뮬레이션의 결과를 보인다.



(그림 14) 클러스터 헤드 선출 기법별 클러스터 헤드 변경 횟수 비교



(그림 15) 클러스터 헤드 선출 기법별 클러스터 병합 및 분할 횟수 비교

그래프에서 Constant로 나타낸 부분은 클러스터 병합을 요청받는 클러스터의 헤드를 클러스터 헤드로 선출하는 기법을 나타낸 것이고, Random은 클러스터 내부의 임의의 노드를 클러스터 헤드로 선출하는 방식을 나타낸 것이다. 전체적인 성능은 제안 기법이 가장 우수한 것으로 나타났고, 병합을 요청받는 클러스터의 클러스터 헤드를 선출하는 방법이 그 뒤를 이었으며, 임의로 클러스터 헤드를 선출하는 기법은 가장 나쁜 성능을 나타냈다. 하지만 클러스터 병합과 분할 작업 횟수에 있어서는 임의로 클러스터의 헤드를 선출하는 기법이 제안 기법보다 성능이 좋은 경우가 나타났다. 또한 노드의 이동 속도가 증가함에 따라서 성능 개선의 정도가 줄어드는 경향을 나타내었다.

이와 같이 임의로 클러스터의 헤드를 선출하는 기법이 전체적인 성능이 떨어지면서도 병합과 분할 작업횟수가 적게 나타나는 이유는 클러스터의 병합이나 분할에 따른 클러스터 헤드 선출보다 클러스터의 분열에 따른 클러스터 헤드 선출이 증가하였기 때문이다. 클러스터를 임의로 선출하게 되면 클러스터의 가장자리에 클러스터 헤드가 자리 잡을 가능성이 커진다. 이렇게 되면 클러스터 분열에 따른 클러스터 헤드 선출이 증가한다. 하지만 클러스터가 분열했을 때 클러스터 헤드를 가지지 못한 클러스터의 크기가 상대적으로 커지기 때문에 병합과 분할 횟수는 기존 기법보다 줄어든다.

시뮬레이션 결과는 노드의 이동 속도가 증가함에 따라서 성능 개선의 정도가 줄어드는 현상을 보인다. 이는 노드의 이동 속도가 증가함에 따라 클러스터 헤드 선출 기법이 클러스터 유지 오버헤드에 미치는 영향이 감소하기 때문이다. 클러스터 헤드의 변경에 영향을 미치는 요인은 노드의 이동에 따른 분열 횟수, 새로 선출된 클러스터 헤드의 위치, 클러스터의 크기 등이 있다. 클러스터의 크기가 고정된 상태에서 노드의 이동 속도가 증가하면 분열 횟수가 증가하므로 클러스터 헤드의 위치를 결정짓는 클러스터 헤드 선출 기법이 차지하는 비중이 상대적으로 감소하게 된다. 따라서 노드의 이동 속도가 증가하면 클러스터 헤드 선출 알고리즘에 따른 성능 개선 효과도 감소하게 된다.

제안한 클러스터 헤드 선출 기법은 클러스터의 토폴로지 정보를 유지하는 Cluster Topology List를 필요로 한다. Random 또는 Constant 기법에서는 Cluster Topology List를 필요로 하지는 않지만, 클러스터의 분열을 감지하기 위해서 Cluster Topology List를 유지해야 하므로 Random이나 Constant 기법에 비해 제안 기법은 추가적인 오버헤드를 발생 시키지 않는다.

5.5 다른 클러스터링 기법과의 비교

이 절에서는 기존에 존재하는 로드 밸런싱 클러스터링 기법과 제안 기법을 비교한다. <표 3>은 로드 밸런싱 클러스터링 기법 간의 차이점을 보인다.

〈표 3〉 로드 밸런싱 클러스터링 기법 비교

성능지표 / 기법	멀티홉 지원	Ripple effect	클러스터 상태 유지	클러스터 헤드 선출 기준
AMC	Yes	No	No	클러스터 크기
DLBC	No	Yes	No	N/A
제안 기법	Yes	No	Yes	이웃 클러스터 멤버의 수

DLBC는 멀티홉 클러스터링을 지원하지 않으며, 클러스터링 헤드가 변화되었을 때 다른 클러스터에 영향을 미치는 ripple effect가 발생한다. 반면 AMC와 제안 기법은 멀티홉 클러스터링을 지원할 뿐만 아니라 ripple effect도 발생하지 않는다. 하지만 AMC에서는 클러스터 상태가 아닌 노드의 상태만을 유지하므로, 클러스터 헤드를 가지지 못하는 클러스터가 발생할 수 있다.

본 논문에서 제안된 제안 기법은 멀티홉 클러스터링을 지원하며 ripple effect가 발생하지 않는다. 또한 각 노드가 자신이 속한 클러스터의 상태와 토폴로지를 유지함으로써, 클러스터 토폴로지 변화에 대해 빠르고 정확하게 대응할 수 있다. 그리고 클러스터 헤드를 선출 할 때, 매번 가장 높은 차수의 노드를 선출함으로써, 클러스터 헤드의 이탈로 인한 클러스터 헤드의 재선출을 줄일 수 있다.

6. 결론 및 향후 연구

다중 홉 클러스터링을 지원하고, 클러스터 헤드 간의 로드 밸런싱을 고려한 클러스터링 기법을 제안하였다. 제안 기법은 노드의 이동성에 관계없이 클러스터 크기의 임계치 범위의 클러스터 크기를 일정하게 유지하고, 이와 같은 안정성을 제공하기 위해 클러스터를 구성하는 노드들은 클러스터 헤드 선출과 클러스터의 병합 및 분할 작업을 계속해서 수행한다. 이동성이 증가하고 클러스터의 크기가 작아질수록 클러터를 유지하는 오버헤드가 증가하는 경향을 나타내었다. 특히, 클러스터의 크기가 커짐에 따라 클러스터를 유지하기 위한 오버헤드가 줄어드는 현상은 멀티홉 클러스터링의 장점을 잘 보여준다.

향후 연구 과제로는 이동성을 고려한 클러스터 멤버의 선정 기법이 필요하다. 본 제안 기법은 이동성에 따른 오버헤드 증가가 급격하게 일어났다. 이는 클러스터 멤버를 선정할 때 이동성을 고려하지 않았기 때문이다. 이동 노드의 이동 방향 및 속도를 기반으로 클러스터 멤버를 선정한다면

보다 효율적인 클러스터링 기법이 될 것으로 기대된다.

참 고 문 헌

[1] M.S. Corson and J. Macker, "Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations," RFC 2501, Internet Engineering Task Force, Jan., 1999.

[2] A. Nasipuri and S.R. Das, "On-Demand Multipath Routing for Mobile Ad Hoc Networks," Proc. of IEEE ICCCN'99, Boston, MA, pp.64-70, Oct., 1999.

[3] J.Y. Yu and P.H.J. Chong, "A survey of clustering schemes for mobile ad hoc networks," IEEE Communication Survey and Tutorial, Vol.7, No.1, pp.32-48, Jan., 2005.

[4] S. Basagni, "Distributed Clustering for Ad Hoc Networks," Proc. of Int'l Symposium on Parallel Architectures, Algorithms and Networks, pp.310-315, June, 1999.

[5] J. Wu and H. L. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," Proc. of 3rd Int'l. Wksp. Discrete Algorithms and Methods for Mobile Comp. and Commun., 1999, pp.7-14.

[6] Y. Chen and A. Liestman, "Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks," Proc. of 3rd ACM Int'l. Symp. Mobile Ad Hoc Net. & Comp., pp.165-172, June, 2002.

[7] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," Proc. of IEEE SICON'97, pp.197-211, Apr., 1997.

[8] J. Y. Yu and P. H. J. Chong, "3hBAC(3-hop between Adjacent Clusterheads): a Novel Non-overlapping Clustering Algorithm for Mobile Ad Hoc Networks," Proc. of IEEE Pacrim'03, Vol.1, pp.318-321, Aug., 2003.

[9] P. Basu, N. Khan, and T. D. C. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," Proc. of IEEE ICDCSW'01, pp. 413-418, Apr., 2001.

[10] T. Ohta, S. Inoue, and Y. Kakuda, "An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks," Proc. of 7th IEEE Int'l Symp. on Autonomous Decentralized Systems(ISADS2003), pp.293-300, Apr., 2003.

[11] A. D. Amis and R. Prakash, "Load-Balancing Clusters in Wireless Ad Hoc Networks," Proc. of 3rd IEEE ASSET'00, pp. 25-32, Mar., 2000.

[12] E. Gafni and D. Bertsekas. "Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology," IEEE Transactions on Communications, pp.11-18, Jan., 1981.

임 원 택

e-mail : imcliff@ece.skku.ac.kr

1995년 2월 성균관대학교 정보통신공학부(학사)

2005년 3월~현재 성균관대 대학원 정보통공학부 석사과정

관심분야 : 이동 애드혹 네트워크, 이동 에이전트

김 구 수

e-mail : gusukim@ece.skku.ac.kr

1994년 2월 성균관대학교 정보공학과(학사)

1996년 2월 성균관대학교 대학원 정보공학과(석사)

2006년 2월 성균관대학교 대학원 전기전자및컴퓨터공학부(박사)

현 재 성균관대학교 대학원 정보통신공학부 연구원

관심분야 : 분산컴퓨팅, 이동 에이전트, 유비쿼터스 미들웨어

김 문 정

e-mail : tops@korea.ac.kr

1998년 2월 성균관대학교 전기전자및컴퓨터공학부(학사)

2000년 2월 성균관대학교 대학원 전기전자및컴퓨터공학부(공학석사)

2005년 2월 성균관대학교 대학원 전기전자및컴퓨터공학부(공학박사)

현 재 고려대학교 정보보호기술센터(CIST) 정보보호대학원 연구원

관심분야 : 이동 컴퓨팅, 무선 ad-hoc 네트워크, 보안 라우팅 프로토콜, 유비쿼터스 컴퓨팅

엄 영 익

e-mail : yieom@ece.skku.ac.kr

1983년 서울대학교 계산통계학과(학사)

1985년 서울대학교 대학원 전산과학과(이학석사)

1991년 서울대학교 대학원 전산과학과(이학박사)

2000년~2001년 Dept. of Info. and Comp. Science at UCI 방문교수

2005년 한국정보처리학회 편집위원장

현 재 성균관대학교 정보통신공학부 교수

관심분야 : 유비쿼터스 컴퓨팅, 분산 컴퓨팅, 이동 에이전트, 시스템 소프트웨어, 시스템 보안 등