애드 혹 네트워크에서 링크밀도기반 클러스터 구축을 이용한 효율적인 플러딩

이 재 현[†] · 권 경 희[††]

요 약

플러딩 방식은 중복 패킷 전송 과 패킷 충돌 같은 근본적인 문제점을 가지고 있으면서도 무선 애드 혹 네트워크에서 임의의 싱크 노드에 대한 경로를 찾기 위해 자주 사용된다. 플러딩 방식이 가지고 있는 이러한 문제점을 개선하기 위한 방법들 중의 하나로 클러스터 플러딩 기법이 제안되어 사용되고 있다. 본 논문에서는 임의의 노드로부터 통신범위 안에 있는 노드들의 수인 노드들의 밀집도 이용하여 헤더를 선출하는 밀도기반 클러스터 플러딩 기법을 제안한다. 네트워크에서 발생하는 중복 패킷전송 및 패킷충돌과 같은 추가비용을 감소하는 방법은 플러딩을 하지 않는 비 플러딩(non-flooding) 노드를 가능한 한 많이 만드는 것이고, 이를 위해서는 가능한 많은 노드들을 멤버로 가지고 있는 클러스터 헤더를 선출하는 것이다. 제안한 방식은 신뢰할 수 있는 네트워크를 유지하고, 네트워크 트래픽의 효율성을 증가시킬 것이다. 본 논문에서는 NS2를 이용한 시뮬레이션을 통하여 기존의 클러스터 방식에 비해 밀도 기반 클러스터가 비 플러딩 노드의 수를 증가시켜 네트워크의 성능저하 없이 네트워크 트래픽의 효율성이 향상되는 것을 확인한다.

키워드 : 클러스터링, 애드 혹 네트워크, 플러딩, 유비쿼터스

Efficient Flooding in Ad hoc Networks using Cluster Formation based on Link Density

Jae-Hyun Lee[†] · Kyung-Hee Kwon[††]

ABSTRACT

Although flooding has the disadvantages like a transmission of duplicated packets and a packet collision, it has been used frequently to find a path between a source and a sink node in a wireless ad hoc network. Clustering is one of the techniques that have been proposed to overcome those disadvantages. In this paper, we propose a new flooding mechanism in ad hoc networks using cluster formation based on the link density which means the number of neighbors within a node's radio reach. To reduce traffic overhead in the cluster is to make the number of non-flooding nodes as large as possible. Therefore, a node with the most links in a cluster will be elected as cluster header. This method will reduce the network traffic overhead with a reliable network performance. Simulation results using NS2 show that cluster formation based on the link density can reduce redundant flooding without loss of network performance.

Key Words : Clustering, Ad hoc Network, Flooding, Ubiquitous

1. 서 론

플러딩은 중복패킷 전송 및 패킷충돌과 같은 단점을 가지고 있는 기법이다. 그러나 이러한 단점에도 플러딩을 이용한 라우팅 방식은 싱크 노드까지 최단 경로를 발견할 수 있다는 장점과 설정된 경로가 중간에 단절되었더라도, 싱크 노드가 물리적으로 연결이 분리되지 않은 상태라면 다른 우회경로를 찾아 연결설정을 해준다는 장점을 가지고 있다. 이러한 장점은 실제 응용분야에 적용할 때 네트워크의 안정성 부분에 있어 매우 중요한 요소이다.

플러딩을 응용한 분야로는 전시회, 야외캠핑, 전쟁터 등 스스로 네트워크를 구성하고 운영될 수 있는 MANET(Mobile Ad hoc NETwork) 기술을 기반으로 하는 분야에 적용되고 있다. 또한 현재의 플러딩 기법을 확장한다면 홈 네트워크 및 빌딩 오토메이션 시스템 등 향후 유비쿼터스 시스템에서도 다양하게 활용할 수 있을 것이다. 따라서 이러한 응용 서비스를 확장해 나가기 위해서는 기존이 플러딩이 가지고 있는 단점을 극복하는 연구가 수반되어야 한다.

플러딩을 효율적으로 하기 위한 연구들이 노드의 위치정보 및 이웃노드의 정보를 이용하여 비 플러딩 노드 수를 증가시키는 방법들을 제안하였다. 또 다른 연구방향으로는 여

† 준 회 원 : 단국대학교 컴퓨터과학과 박사수료
†† 종신회원 : 단국대학교 컴퓨터과학과 교수
논문접수 : 2007년 10월 16일, 심사완료 : 2007년 12월 12일

러 개의 노드를 하나로 묶는 클러스터링 방식이 제안되고 있다. 클러스터링 방식에서의 노드들은 *헤더, 게이트웨이, 비 플러딩, 초기상태*와 같은 노드상태를 가지고 있고, 각 노드들은 그 상태에 따라 자신의 역할을 수행한다. 클러스터링 방식의 장점은 여러 개의 노드들을 하나의 클러스터로 만들어 클러스터내의 헤더와 게이트웨이 노드들을 제외한 노드들이 비 플러딩 노드가 되게 할 수 있다는 것이다. 여기서 클러스터 헤더는 이웃노드와 1홉 거리에 있어야 하며, 게이트웨이는 2개 이상의 클러스터 헤더 사이에 존재하는 노드를 말한다. 이런 클러스터링 방식의 효과를 최대로 하기 위해서는 전체 네트워크에서 클러스터의 수는 작을수록, 클러스터를 형성하는 노드의 수는 많을수록 좋을 것이다. 이를 위해서는 클러스터 헤더의 선출이 중요한 요소가 된다. 기존의 클러스터링 방식[8][10]에서 클러스터 헤더는 이웃노드의 상태정보만을 통하여 결정되고, 이웃노드의 수와는 무관하게 결정된다. 이것은 더 많은 이웃노드를 갖는 헤더가 클러스터의 헤더로 선출된다는 것을 보장하지 못한다.

본 논문에서는 기존 클러스터링 방식을 좀 더 효율적으로 만들기 위해 밀도기반 클러스터 방식을 제안하고자 한다. 밀도기반 클러스터링은 많은 이웃노드를 갖는 노드가 클러스터 헤더가 되게 하여 비 플러딩 노드의 수를 기존 방식보다 더 증가시킬 수 있게 하였다. 본 논문에서 *진행중인 패킷(On-going packet)*이란 소스에서 싱크 노드까지 전송되는 패킷을 말한다. 헤더의 후보결정은 *진행중인 패킷*을 이용하여 결정하며, 헤더의 선출은 제어패킷을 이용하여 이웃노드의 링크 수와 자신의 링크 수를 비교하여 자신이 헤더가 될 수 있는 지를 결정한다. 제어패킷의 수는 네트워크 구성의 초반에 대부분이 발생하며, 이후에는 헤더의 변경시에만 발생하므로 전체 네트워크 트래픽에는 큰 영향을 미치지 않는다. 이와 같은 방법은 클러스터 헤더가 되는 조건을 타당하고 명확하게 하였고, 또한 헤더가 아닌 노드들의 상태는 이웃노드의 상태를 통하여 결정하도록 하였다. 노드들의 상태 정보는 *진행중인 패킷*에 정보를 삽입하여 이용하였고, 각 노드의 연결정보는 링크 레이어와 *진행중인 패킷*을 통하여 수집하도록 하였다.

본 논문의 구성은, 2장에서 기존에 연구된 플러딩 방식에 대해서 소개한다. 3장에서는 밀도기반 클러스터 구성방법을 제안한다. 4장에서는 밀도기반 클러스터 형성 방식이 중복 패킷 전송을 어느 정도 감소시킬 수 있는지 이동속도와 실험영역에 대한 다양한 상황을 설정하여 시뮬레이션하며 이를 통하여 본 논문에서 제안하는 방식의 효율성을 검증한다. 마지막으로 5장에서는 본 연구에서 제안한 방식에 대한 성과를 논의하며, 향후 추가로 연구할 과제에 대해서 작성하였다.

2. 관련 연구

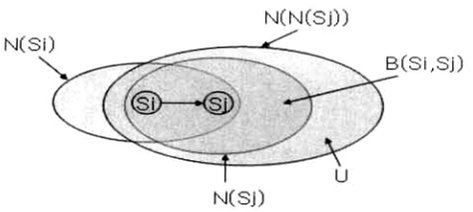
무선 애드 혹 네트워크 환경에서 플러딩에 관한 연구들[1][2][3][4]은 기존의 맹목적 플러딩에 대한 문제점을 다루었고, 그 문제를 해결하기 위해 노력하고 있다. 그러나 무선 애드 혹 네트워크에서 전송비용을 최소로 하는 전송노드들의 집합을 찾는 것이 NP-complete임이 증명되었다[1]. 따라서 효율적인 플러딩 방식에 대한 모든 연구들은 전송 오버헤드를 최소화할 수 있는 경험적 지식을 찾는 것을 목표로 하고 있다[8]. 효율적인 플러딩을 방법들로는,

- 랜덤 하게 결정된 확률 값에 의해 패킷이 재전송 될지를 결정하는 확률적 방법.
- 중복 수신한 패킷의 수를 카운트하여 임계 값과 비교한 후, 재전송 여부를 결정하는 카운터 기반의 방식.
- 노드간의 상대적인 거리를 이용하여 재전송을 결정하는 거리기반 방식.
- 이웃노드들의 위치정보를 미리 수집하여 재전송을 결정하는 위치정보기반 방식.
- 클러스터 헤더 와 게이트웨이의 상태를 갖는 노드들만 전송에 참여하는 클러스터링 방식.

등이 제안되고 있다[1][6]. 위 방법들 중 본 논문에서 연구한 분야는 클러스터링 방식이다. 기존 애드 혹 네트워크에서의 클러스터링 방식은 hierarchical routing schemes [5][12], master selection algorithms[16], power control[5], reliable broadcast[17] 그리고 efficient broadcast[1][18]를 위해 폭넓게 연구되고 사용되어 왔다. 그러나, 클러스터을 이용한 플러딩 방식은 효과적으로 사용될 수 있음에도 주기적인 패킷교환이 필요하다는 이유 때문에 일반적으로 사용을 하지 않았다[8]. 아래 처음 소개되는 방식은 확률적 플러딩 방식이다. 그리고 다음 두 가지 방식은 이웃노드의 정보를 수집하여 재전송을 결정하는 방식으로 수신자가지치기(self-prunin g)와 송신자 가지치기(dominant-pruning) 방식을 소개하였고, 세 번째로는 기존 클러스터링 방식의 문제점인 주기적인 패킷 교환 문제를 해결한 유사한 방식의 두 가지 클러스터링 방식을 소개한다.

첫 번째는 확률적 플러딩 방법이다. 이 방법은 패킷의 도착 율을 100%로 유지하고, 각 노드의 방송으로 인하여 과도하게 발생하는 도착 노드의 중복 수신율을 낮추기 위해 각 노드가 확률 값에 의해 방송여부를 결정하는 방법이다. 확률 값 계산은 각 노드들이 밀집된 지역에서 과도한 방송패킷이 발생한다는 것을 이용하여 노드의 밀도가 높은 지역에서는 방송확률을 낮추고, 밀도가 낮은 지역에서는 방송확률을 높이는 동적인 방법으로 이루어진다[19]. 그리고 [20]은 확률적 플러딩 방법에서 이용한 밀도를 트래픽 개선이 아닌 대체경로를 구하는데 사용한다. 그러나 [19]에서 제안한 방법은 각 노드의 밀도계산을 위해 2홉 이내의 이웃노드 정보를 확보해야 하며, 노드의 이동성을 고려하고 있지 않다는 점에서 MANET (Mobile Ad hoc Network)환경에는 적합하지 않다.

두 번째 알고리즘은 수신자 가지치기로서, 임의의 노드 A는 자신과 인접한 노드의 리스트 N(A)를 전송 패킷에 같이 보낸다. 인접한 B노드가 이 패킷을 받은 후 아래 공식을



(그림 1) 송신자 가지치기

만족하면 주위에 패킷을 전달하고, 그렇지 않으면 전달하지 않는다 [4].

공식: N(B) - N(A) - {A} != 공집합

세 번째 알고리즘은 송신자 가지치기이다. 노드 S의 인접 노드를 N(S)라 가정한다.

Sj가 Si에게 패킷을 전송 받았고, 노드 Sj로부터 2홉 안에 있는 노드들을 N(N(Sj))라 가정한다. 이 노드들 중 Sj, Si, N(Si)는 이미 패킷을 받았고, Sj는 2홉 이내의 모든 노드가 패킷을 받을 수 있도록 해야 한다. 또한Sj는 전송횟수를 줄이기 위해 forward list를 최소화 해야 한다.

Sj는 U=N(N(Si))-N(Si)-N(Sj)에 포함된 모든 노드가 패킷을 받도록 forward list를 결정하면 된다.

N(N(Si))노드들은 Si가 패킷을 전송할 때 패킷을 전송 받음을 보장받았으므로 B(Si, Sj) = N(Sj)-N(Si)에 속하는 노드들 중에서 forward list를 결정하면 된다. B(Si, Sj)를 {b1, b2, .. , bn}라고 하면,

$$\bigcup_{b \in F} (N(b) \cap U) = U$$

가 되도록 F⊆B(Si, Sj)를 결정하여야 한다.

위에서 가장 작은 F를 찾는 것은 greedy set cover 알고리즘을 이용한다[4][11].

네 번째는 클러스터를 이용한 방식으로 다음 두 가지의 유사한 연구가 있다. 하나는 수동적인 클러스터링 방식이고[8], 다른 하나는 플러딩 노드의 수를 감소시키기 위한 유니케스트 기반의 클러스터링 방식이다[10]. 수동적인 클러스터링 방식은 6가지의 상태를 가지며 이웃노드의 상태에 따라 자신의 상태를 결정한다. 각 상태는 *초기상태, 클러스터 헤더, 게이트웨이, 게이트웨이 준비, 분산 게이트웨이, 비 플러딩* 노드로 이루어 진다. 위의 방식은 클러스터 헤더를 선출하고, 중복 게이트웨이를 제거하여 플러딩 노드를 감소시키는 방식을 제안한다. 그러나 비 플러딩 노드(ORDINARY_NODE) 상태가 되려면 충분히 이웃하는 게이트웨이들을 학습한 후에 가능하므로 잦은 방송 메시지의 충돌이 발생한다[8].

반면, 유니케스트 기반의 클러스터링 방식은 수동적인 클러스터링 방식의 단점을 보완하기 위하여 노드의 상태를 6단계에서 4단계로 간소화하였고, 이웃 노드들과의 통신방식을 방송이 아닌 유니케스트 방식을 사용하였다. 그리고 이 방식에서 클러스터 헤더 또는 게이트웨이가 되기 위해서는 비 플러딩 노드 상태가 아닌 노드가 게이트웨이 또는 클러스터 헤더 노드로부터 패킷을 수신해야 한다. 그리고 비 플러딩노드 상태가 되기 위해서는 게이트웨이나 클러스터 헤더 상태인 노드가 같은 선행자 식별자 및 클러스터 상태를 가지고 있으면서 더 작은 식별자를 가진 노드를 발견하면 된다[10]. 유니케스트 기반의 클러스터링 방식은 기존에 제안된 논문들에서 사용하는 클러스터 헤더와 게이트웨이의 개념을 다르게 사용하고 있다. [10]번에서 사용하는 클러스터 헤더는 멤버노드를 갖는 방식이 아닌 전송노드를 표시하기 위함이고 게이트웨이는 클러스터 헤더의 이웃 노드들 중에서 노드의 식별자로써 구분되는 방식이다.

위의 두 클러스터링 방식은 각 노드 사이에 정해진 규칙을 통하여 노드 자신의 상태 값을 변경하며 클러스터를 유지한다. 그리고 이웃노드에 대한 정보를 수집하는 방식에 있어서도 혼합모드(promiscuous)를 사용하는 것으로 동일하다.

본 논문에서 제안하는 방식은 위에서 제안하는 방식의 조건은 유사하지만, 클러스터 헤더의 선출하는 방식에서 기존의 방식과 차별을 두었다. 기존의 클러스터 헤더의 선출방식은 효율적인 헤더 선출보다는 클러스터 구축을 위한 헤더 선출방법을 제시하지만, 헤더 선출방식의 효율성을 보장할 수 없다. 본 논문에서는 클러스터 헤더 선출 시, 최대한 이웃노드가 많은 노드를 클러스터 헤더로 선출하여 기존의 클러스터 구성 방식[10]보다 많은 비 플러딩 노드를 생성할 수 있음을 보인다.

3. 밀도기반 클러스터링

본 장에서는 밀도기반 클러스터를 구성하는 방법에 대해서 설명한다. 우선 밀도기반 클러스터링의 개요 및 노드들의 상태에 대해 설명한 후, 클러스터 형성에 관하여 알아보고 마지막으로 예를 통해 밀도기반 클러스터 형성 및 노드 이동 후 클러스터의 변화과정을 살펴보겠다. 설명의 편의를 위해 다음과 같이 패킷을 정의하여 사용하도록 하겠다.

O_Packet: Init상태의 노드가 일반노드가 된 후, 전송하는 패킷으로 자신이 일반노드임을 알린다. 이 O_Packet을 수신한 노드들은 자신들이 CH가 될 수 있는지를 점검하도록 한다. 만약 O_Packet을 수신한 노드 중에 CH가 발생하면 O_Packet을 전송한 노드는 게이트웨이 노드가 되어 두 CH 노드를 연결할 것이다.

CH_Packet: 자신이 클러스터 헤더 임을 알리는 패킷으로 비 플러딩 노드를 생성하는 역할을 한다.

3.1 밀도기반 클러스터링 개요

밀도기반 클러스터링에서 밀도란 하나의 노드가 이웃노드와 연결된 링크의 수를 말한다. 이 링크의 수가 많을수록 밀도가 높다고 말하며, 클러스터 헤더가 될 확률이 높게 된다. 클러스터 헤더가 선출되면 이웃노드들은 *진행중인 패킷*을 통하여 상태를 결정하게 된다. 본 논문에서 각 노드의 상태는 6가지이며 다음과 같이 정의한다.

- 초기상태(INIT): 노드가 클러스터 구성에 참여하지 않은 상태를 말하며, 이 상태는 네트워크 구성 시작일 때 또는 비정상적으로 이웃노드와의 연결이 끊어졌을 때 갖게 된다.
- 클러스터 헤더 준비(CHR): 초기상태의 노드가 O_Packet을 수신하면 이 상태가 된다. 이 상태는 클러스터 헤더가 될 수 있는 후보의 자격을 갖게 되는 것이며, 이웃노드와의 밀도 비교를 통해 클러스터 헤더로 발전하거나 일반노드가 될 수 있다.
- 클러스터 헤더(CH): 클러스터 헤더 준비상태에 있는 노드가 같은 상태의 이웃노드들과 밀도를 비교한 후 밀도가 가장 높은 노드가 이 상태가 된다. 또는 클러스터 헤더 준비상태의 노드가 밀도가 높은 이웃노드로부터 일정시간 CH_Packet을 수신하지 못하면 자신을 클러스터 헤더 상태로 변경한다. 클러스터 헤더는 수신한 패킷을 자신의 이웃노드들에게 방송한다.
- 게이트웨이(GW): 두 개 이상의 클러스터 헤더를 이웃노드로 가지고 있을 때 이 상태가 된다. 게이트웨이 노드는 클러스터와 클러스터간의 연결을 담당하는 노드로서 클러스터 헤더로 부터 도착한 패킷을 다른 클러스터 헤더로 전달한다.
- 노드게이트웨이(NGW): 일반노드(ORDIARY)가 초기 상태의 단말노드로부터 INIT_Packet을 받으면 자신을 이 상태로 만든다. 노드 게이트웨이는 자신과 연결된 초기상태의 노드가 터미널 노드일 경우 그 노드에게 패킷을 전송하기 위한 노드로서 게이트웨이 노드와 구분된다.
- 일반노드(ORDINARY): 비 플러딩 노드라 부르며, 이웃에 하나의 클러스터 헤더가 있으면 이 상태가 된다. 또는 자신이 단말노드이며 이웃 노드가 노드 게이트웨이 상태이면 이 상태가 된다. 일반 노드는 패킷을 수신만 하며 다른 노드로 재전송하지 않는다.

각 노드들은 *진행중인패킷*을 이용하여 자신의 상태 정보를 이웃에게 전달하고 이 패킷을 수신한 노드들은 자신의 상태변화가 필요한지를 확인한다. 본 논문에서는 클러스터 구성 및 자신의 상태변화를 위해서 3 종류의 제어패킷을 추가로 사용한다.

첫 번째는 CHR_Packet으로 클러스터 헤더 준비상태의 노드들이 헤더 경합을 하기 위해 이웃간 전송하는 패킷이다. 두 번째는 CHChange_Packet으로 기존의 헤더가 다른 헤더 영역으로 이동하여 클러스터가 해체될 경우 자신의 멤버들에게 이 패킷을 보내 노드들의 상태를 새롭게 하도록 한다. 세 번째는 INIT_Packet으로 단말노드가 자신이 단말노드임을 확인하고 자신에게 패킷을 전달한 노드에게 이 패킷을 전송한다. 전송 후, INIT_Packet을 전송한 노드는 일반노드 상태로 변경하고, 이 패킷을 받은 노드는 노드 게이트웨이 상태로 변경한다.

3.2 밀도기반 클러스터 구성

본 논문에서 제안하는 클러스터의 구성 다음과 같은 순서로 이루어진다.

① 소스 전송, 최초의 소스는 클러스터 헤더가 된다.

② 소스에 인접한 노드들은 소스로부터 CH_Packet을 수신 후 자신의 상태를 일반노드 상태로 변경 후, 그 패킷에 자신의 상태를 실어 O_Packet을 전송한다.

③ 일반노드로부터 O_Packet을 수신한 Init 상태의 노드들은 자신을 CHR(클러스터 헤더 준비) 상태로 만들고 CHR_Packet을 전송한다. 이웃하는 CHR 노드들은 밀도를 비교하여 밀도가 가장 높은 노드가 자신의 상태를 클러스터 헤더로 만들고,CH_Packet을 전송한다. 인접한 노드들은 패킷수신 후 자신들의 상태를 일반노드 상태로 변경하고 O_Packet을 전송한다. 이후 3)의 상황을 반복한다.

④ 위 3)의 상황 중에 클러스터 헤더로부터 2홉 떨어진 CHR 노드들은 일정시간(0.1s) 경과 후 CH_Packet을 수신하지 못하면 자신의 상태를 클러스터 헤더로 바꾸고 CH_Packet을 전송한다. 역시 이 CH_Packet을 수신한 Init 노드들은 자신의 상태를 일반노드로 변경 후, O_Packet을 전송한다.

⑤ 위 3)의 상황 중에 단말노드가 클러스터 헤더를 가지지 못하고 일반노드와 이웃이 될 경우에, INIT_Packet을 전송하게 된다. 그러면 자신의 상태는 일반노드가 되며, 자신의 이웃인 일반노드는 노드 게이트웨이 상태의 노드가 된다.

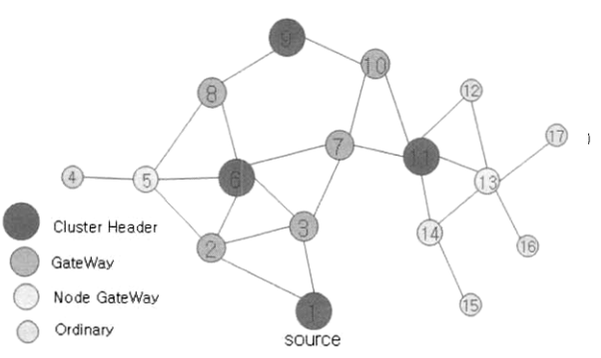
⑥ 위의 과정 중에 하나 이상의 클러스터 헤더를 이웃으로 가지게 되는 노드들은 게이트웨이 상태의 노드가 된다.

위의 1)~6)까지의 과정을 거치고 나면 밀도를 기반으로 하는 클러스터가 구축된다. 이와 같은 방법으로 선출된 클러스터 헤더는 기존보다 많은 멤버노드를 가지게 될 것이다. 하나의 클러스터 헤더에 많은 멤버가 존재한다는 것은 그렇지 않은 경우에 비해 더 많은 비 플러딩 노드를 생성한다는 것을 의미한다. 다음은 임의의 네트워크에서 밀도기반 클러스터가 형성되는 방법에 대해 설명하겠다.

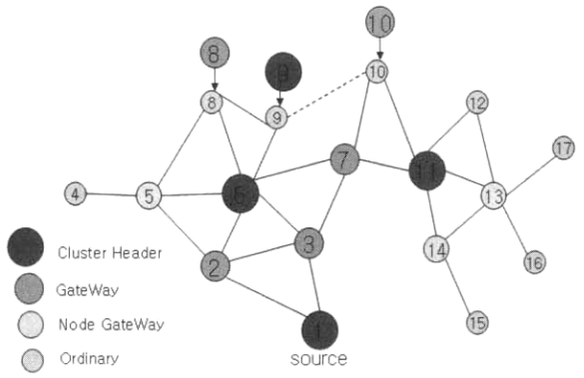
3.3 밀도기반 클러스터 형성 예

아래의 (그림 2)는 본 논문에서 제안하는 밀도기반 클러스터를 구축한 결과이다. 절차는 다음과 같다.

1번 노드가 소스(클러스터 헤더)가 되어 최초 CH_Packet을 전송하면 2, 3번 노드는 자신의 상태를 일반노드로 변경



(그림 2) 밀도기반 클러스터



(그림 3) 노드 이동 후 클러스터 변경

하고 O_Packet을 전송한다. Init상태의 5, 6, 7 번 노드는 O_Packet을 수신 후 자신들의 상태를 클러스터 헤더 준비 상태로 변경하고, CHR_Packet을 전송할 것이다. 이 중 6번 노드가 밀도가 가장 높으므로 6번 노드는 CH_Packet을 전송하게 될 것이고, 이 패킷을 수신한 노드들은 자신의 상태를 변경할 것이다. 2, 3번은 2개 이상의 클러스터 헤더를 가지므로 게이트웨이 상태가 되고, 5, 7, 8 번은 일반노드로 자신의 상태를 수정한 후 다시 O_Packet을 전송할 것이다. 9, 10, 11번 노드는 이전과 동일한 방법에 의해서 자신의 상태를 변경해 갈 것이다. 4번 노드는 5번 노드가 보내온 O_Packet을 수신 후 자신이 단말노드임을 확인하고 5번 노드에게 INIT_Packet을 전송한 후 자신의 상태를 일반노드로 변경한다. INIT_Packet을 수신한 5번 노드는 자신의 상태를 노드 게이트웨이 상태로 변경한다.

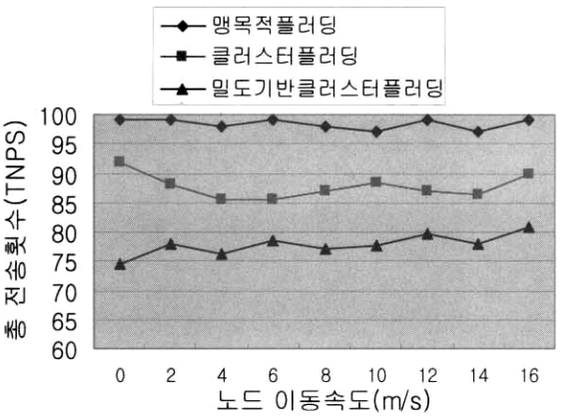
(그림 3)은 9번인 클러스터 헤더 노드가 이동하여 6번 클러스터 헤더와 인접하게 되었을 때 각 노드의 변경 상태를 나타내는 것이다. 9번 노드가 이동하여 6번 노드와 인접하게 되었을 때 9번 노드는 6번 노드로부터 *전송중인 패킷*을 수신하게 될 것이고, 수신한 패킷에서 6번 노드가 자신과 같은 상태인 것을 인지할 것이다. 이 때 9번 노드는 6번 노드보다 밀도가 떨어지므로 자신의 클러스터 헤더 상태를 해제하며, CHChange_Packet을 자신의 멤버노드들에게 전송한다. 그리고 자신의 상태는 일반노드로 변경한다.

9번 노드로부터 CHChange_Packet을 수신한 8번과 10번 노드는 수신 후 자신들의 클러스터 헤더의 수가 2에서 1로 감소했으므로 자신들의 상태를 일반노드 상태로 변경하게 된다. 이 때 만약 9번 노드에 붙어있는 일반 노드가 있다면, 그 노드는 Init상태가 됨과 동시에 자신이 단말노드임을 확인하고 9번 노드에게 INIT_Packet을 전송 후 자신의 상태를 일반노드로 변경한다. INIT_Packet을 수신한 9번 노드는 노드 게이트웨이 상태가 될 것이다.

4 시뮬레이션

4.1 시뮬레이션 환경

본 논문에서 제안하는 클러스터 구성방식의 성능을 분석하기 위하여 NS2[14]를 사용하였으며, 본 논문에서 사용한 NS2는 무선 애드 혹 환경에서 이동노드를 시뮬레이션 할 수 있도록 한 Monark Project[13]를 포함하고 있다. 본 논문에서는 기존에 연구된 방법들과 비교하기 위하여 크게 두 가지의 실험 환경을 가지고 시뮬레이션을 수행하였다.



(그림 4) 1000*1000m^2 환경에서 0.25초 간격으로 패킷을 전송했을 때의 총 전송횟수를 이동속도 별로 측정

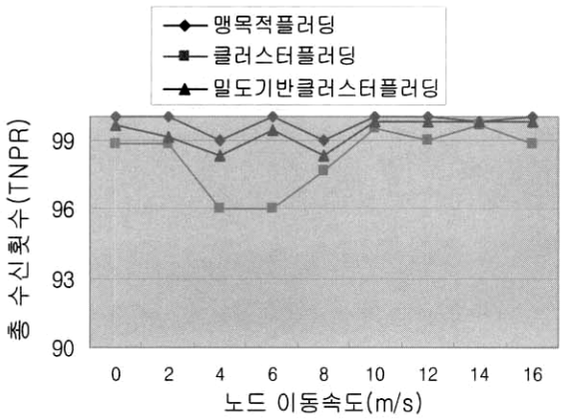
첫 번째는 실험영역을 1000*1000 m^2로 고정한 후, 100개의 이동노드들을 100초의 정지시간을 주며 0m/s부터 16m/s까지 이동시키며 600초(10분)동안 시뮬레이션을 수행하였다. 또한 소스로부터 패킷의 전송 주기는 1/4초를 주었으며, 전송범위는 250m로 고정하였다. 위의 환경에서 시뮬레이션한 결과가 (그림 4) (그림 5)와 같다. 두 번째는 각 노드들을 고정시킨 상황에서 실험영역을 600m^2부터 2000m^2까지 200m^2씩 증가시키면서 실험을 하였다. 두 번째의 경우도 첫 번째와 동일한 실험시간, 전송주기, 전송범위를 가지고 실험하였다. 두 번째 환경에서 시뮬레이션한 결과가 (그림 6) (그림 7)과 같다. 실험결과 그림에 대한 이해를 위해 다음 두 개의 용어를 설명하겠다.

- 총 전송횟수(TNPS: Total number of Packet Sent for one broadcast): 하나의 소스 전송패킷에 대하여 전송에 참여하는 모든 노드의 수. 이를 계산하기 위하여 전송한 모든 패킷 수를 소스가 전송한 패킷의 수로 나누었음.
- 총 수신횟수(TNPR: Total number of Packet Received for one broadcast): 하나의 소스 전송패킷에 대하여 그 패킷을 수신한 모든 노드의 수. 이를 계산하기 위하여 모든 노드의 패킷수신횟수를 소스가 전송한 패킷의 수로 나누었음.

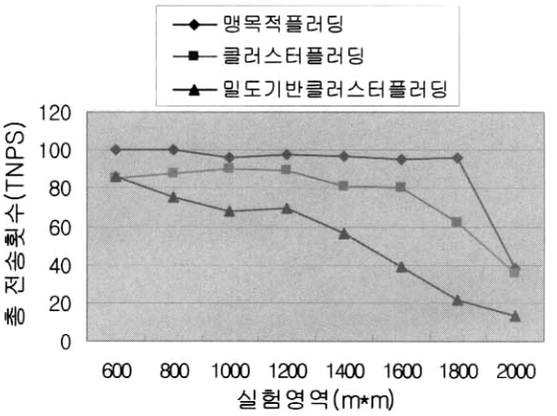
본 실험에서 노드의 좌표와 이동에 대해서 랜덤하게 지정하였으며, 결과 수치는 다수의 실험결과 값을 평균 내어 작성하였다.

4.2 실험결과 분석

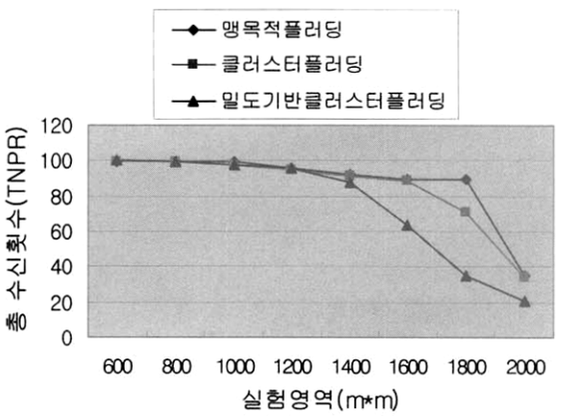
본 논문에서 제안하는 방식의 효율성을 분석하기 위하여 기존에 맹목적 플러딩 방식을 사용하는 방식[6]과 [6]을 개



(그림 5) 1000*1000m^2 환경에서 0.25초 간격으로 패킷을 전송했을 때의 총 수신횟수를 이동속도 별로 측정



(그림 6) 실험영역을 600m^{2}부터 2000m^{2}로 확대해가면서 총 전송패킷의 수를 측정. 노드들은 고정되어 있으며 전송주기는 0.25초로 동일함



(그림 7) 실험영역을 600m^{2}부터 2000m^{2}로 확대해가면서 총 수신패킷의 수를 측정. 노드들은 고정되어 있으며 전송주기는 0.25초로 동일함

선한 기존의 클러스터링 방식[10], 이 두 가지 방식과 비교를 하였다.

(그림 4)에서는 보면 제안한 밀도기반 클러스터링 방식은 맹목적 플러딩 방식보다는 약 22%정도 그리고 기존의 클러스터 플러딩 보다는 평균적으로 약 10% 정도의 패킷전송율 감소 시켰음을 볼 수 있다. 기존의 클러스터 구성방식은 클러스터 구성에 있어서 헤더의 역할과 게이트웨이의 역할이 구분되지 않고 단지 이웃노드의 상태에만 의존하여 모든 노드들이 자신의 상태를 결정하므로 이동속도에 따른 그래프의 규칙을 찾기가 어렵다. 반면 제안한 밀도기반 클러스터링 방식은 헤더의 선출방식이 명확하며 그 규칙을 기반으로 클러스터가 형성됨을 볼 수 있다. (그림 4)을 보면 네트워크에서 노드의 이동속도가 높아질수록 전송패킷의 수가 조금씩 증가함을 볼 수 있다. 증가치를 감안하더라도 기존의 클러스터링 방식보다는 총 전송횟수에서 효율적이라는 것을 알 수 있다. 그러나 총 전송횟수가 낮더라도 총 수신횟수가 같이 낮다면 네트워크 성능이 문제가 될 수 있기 때문에 효율성 분석은 (그림 5)와 함께 이루어져야 한다. (그림 5)를 보면 제안한 방식이 기존의 클러스터링 방식보다 높은 수치의 수신율을 나타내고 있음을 알 수 있다. 그리고 99% 이하로 떨어지는 수치는 노드의 이동으로 인해 이웃노드와의 링크가 잠시 중단되는 노드가 존재하기 때문이다. 따라서 1000 * 1000m^{2}에서 지금과 동일한 이동속도로 계속적인 실험을 한다면 모든 점들이 99%이상으로 수렴할 것이다.

첫 번째 실험의 결과를 통하여 기존의 방식보다 제안한 방식이 전송율 과 수신율을 통해 네트워크 전반에 미치는 트래픽 효율성 및 성능을 보장할 수 있음을 확인하였다.

두 번째 실험은 실험영역의 변화에 따른 고정노드에 대한 실험이다. 실험영역의 변화를 가했을 경우 총 전송횟수의 변화를 (그림 6)에서 보여주고 있다.

(그림 6)에서 보면 밀도기반 클러스터링 방식이 기존의 방식들에 비해 낮은 전송횟수를 가지고 있다는 것을 알 수 있다. 그러나 1400m^{2}를 넘어서면서부터 송신횟수가 현저하게 낮아지고 있다. 따라서 과도한 전송횟수의 감소가 네트워크 성능에 영향을 주는지 확인하기 위해 두 번째 실험에서 (그림 7)의 수신횟수와 같이 네트워크 성능 분석해 보았다. (그림 7)과같이 비교해 보면 1400m^{2}까지는 제안한 방식이 기존의 방식보다 트랙픽 효율성 및 성능에서 효과적인 방식임을 알 수 있다. 그러나 1600m^{2}에서는 제안한 방식의 수신율이 60%까지 떨어지게 되어 기존의 방식보다 낮은 네트워크 성능을 보이고 있다. 그 이후에는 기존의 방식 역시 수신율이 80%를 밑돌아 네트워크 성능에 문제가 발생하게 됨을 알 수 있고, 2000m^{2}에서는 맹목적 플러딩 방식도 같은 문제점을 가지게 된다.

위와 같이 1600m^2 이상의 영역에서는 발생하는 송·수신율의 저하문제는 물리적인 노드간의 연결 실패에 따른 것이다. 그러므로 본 논문에서 제안한 방식이 1600m^2 미만의 영역에서는 기존의 방식보다 네트워크 트래픽에서는 효과적이며, 수신율에서는 유사한 성능을 보이고 있음을 확인할 수 있다.

5. 결 론

본 논문을 통하여 기존의 플러딩 알고리즘에 대해서 살펴보았고, 기존의 플러딩 알고리즘의 문제점인 네트워크 트래픽 오버헤드를 감소하는 방안으로 밀도기반 클러스터링 방식을 제안하였다. 밀도기반 클러스터링 방식의 특징은 클러스터 헤더의 선출방식에 있다. 많은 멤버노드를 갖은 클러스터 헤더가 가장 효율적인 헤더며, 네트워크에 많은 비 플러딩 노드를 만들 수 있다는 가정에서 출발한 것이다.

밀도기반 클러스터 플러딩의 장점은 효율적인 플러딩을 위한 클러스터 방식에서 헤더의 선출문제에 대한 기준을 제시했으며, 그 방식이 기존의 방식보다 효율적임을 실험을 통하여 보였다. 밀도기반 클러스터링 방식은 기존의 클러스터링 방식보다 이동노드 환경에서 약 9.9%의 전송율을 감소시켰고, 정적인 노드에서는 $1400m^2$ 이하 구간에서 약 15.6%의 효율성을 보인다. 밀도기반 클러스터링 방식은 기본 개념을 바탕으로 향후 유비쿼터스 환경에 적합한 서비스 구현을 목표로 하고 있다. 시스템 차원의 구체적인 연구방향을 목표로 할 것이며, 그 응용의 예로써 플러딩을 이용한 홈 네트워크 응용이 될 수 있을 것이다.

참 고 문 헌

[1] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad hoc Network, "in Proc. Of ACM/IEEE International Conference of Mobile Computing and Networking (MOBICOM'99), Sep. 1999, pp.153-167.

[2] Yu-Chee Tseng, Sze-Yao Ni, En-Yu Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Networks", Vol. 1 No. 3, 1995.

[3] Amir Qayyum, Laurent Viennot, Anis Laouiti, Multipoint relaying: "An efficient technique for flooding in mobile wireless networks", INRIA report, March 2000.

[4] H.Lim and C.Kim, "Flooding in wireless ad hoc networks", IEEE computer communications 2000.

[5] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", IEEE Journal on Selected Areas in Communications, Vol. 15, No. 7, Sep.1997, pp. 1265-1275.

[6] Charles Perkins, Elizabeth M. Royer, "Ad Hoc On Demand Distance Vector (AODV) Routing," Internet draft, IETF, Aug. 1998.

[7] Josh Broch, David B. Johnson, David A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet draft, IETF, March 1998.

[8] Y. Yi, M. Gerla, T. Kwon, "Efficient Flooding in Ad Hoc Networks Using On-demand(Passive) cluster Formation,"in Proc. Of 2nd annual Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net 2003), Annapolis, USA, 2003.

[9] Kihwan Kim, Ying Cai, Wallapak Tavanapong, "A Priority Forwarding Technique for Efficient and Fast Flooding in Wireless Ad Hoc Networks", IEEE 2005.

[10] 왕기철, 김태연, 조기환, "애드 혹 네트워크에서 클러스터를 이용한 효율적인 플러딩 방안", 정보과학회논문지, 제32권 제6호, 2005년 12월.

[11] Laszlo Lovasz, "On the ratio of optimal integral and fractonal covers," Discrete Math. 13(1975) pp.383-390.

[12] Krishna, P., Vaidya, N.H., Chatterjee, M., Pradhan, D.K., "A cluster-based approach for routing in dynamic networks," Computer Communication Review, vol.27, (no.2), ACM, April 1997.

[13] CMU Monarch Project: Wireless and Mobility Extensions to ns-2, url: http://www.monarch.cs.rice.edu/cmu-ns.html

[14] Network Simulator 2, UC Berkley. url: http://www.isi.edu/nsnam/ns

[15] "Overview of Intelligent Garget Technology for Life Log Service", C.S.Bae, J.H.Won, D.W.Ryoo, 전자통신동향분석 제21권 제5호, ETRI, 2006년 10월.

[16] Dennis J. Baker and Anthony Ephremides, "The Architectural organization of a mobile radio network via a distributed algorithm," IEEE Transaction on communications, Vol. com-29, No. 11, Nov. 1981.

[17] E.Pagani, G.P. Rossi, Reliable broadcast in mobile multihop packet networks, Mobicom 97 (1997).

[18] M.Jiang, J.Li, Y.C. Tay, "Cluster based routing protocol (CBR) functional specificati -on,"IETF draft 1998.

[19] 김정삼, 류정필, 한기준, "애드 혹 네트워크에서 지역 밀집도에 적응적인 확률적 플러딩 기법", 대한전자공학회논문지TC, 1229-635X, 제42권 제9호 통권339호, pp.11-18, 2005년 9월.

[20] 김상경, 최승식, "노드 밀집도 기반 애드학 라우팅", 한국정보처리학회논문지C, 1598-2858, 제12C권 제4호 통권100호, pp.535-542, 2005년 8월.

이 재 현

e-mail : wogusking@dankook.ac.kr

2000년 단국대학교 전자계산학과 (학사)

2003년 단국대학교 대학원 전자계산학과(이학석사)

2007년 단국대학교 전자계산학과 (박사수료)

관심분야 : ad hoc 네크워크, 홈 네트워크, IPv6, RFID

권 경 희

e-mail : khkwon@dku.edu

1976년 고려대학교 물리학과(학사)

1986년 Old Dominion Univ. Dept. of Comupter Science (M.S)

1992년 Louisiana State Univ. Dept. of Comupter Science (Ph.D.)

1979년~1984년 산업연구원 연구원

1993년~현재 단국대학교 컴퓨터과학과 교수

관심분야 : 컴퓨터 네트워크, 알고리즘 분석 및 설계, 웹 공학