애드 혹 망에서 확장성이 있는 이동 기반 오버레이 멀티캐스트

우부재°, 최영환, 박수창, 이의신, 전야, 박호성, 김상하
충남대학교 컴퓨터공학과
{yufc, yhchoi, winter, eslee, tianye, hspark}@cclab.cnu.ac.kr and shkim@cnu.ac.kr

SLAOM: Scalable Location-Aware Overlay Multicast for Ad hoc Networks

Fucai Yu°, Younghwan Choi, Soochang Park, Euisin Lee, Ye Tian, Hosung Park and Sang-Ha Kim
Dept. of Computer Engineering, Chungnam National University

요 약

이동 애드 혹 망에서 노드들의 이동성은 멀티캐스트를 어렵게 한다. 많은 멀티캐스트 프로토콜에서 데이터 전달 구조 구성과 유지를 하기 위해 주기적인 플러딩이 널리 사용된다. 이런 주기적인 플러딩 메시지는 네트워크의 제어 비용을 심각하게 증가시키고, 에너지와 대역폭 자원을 낭비하게 한다. 데이터 전달 구조 유지 부담을 줄이기 위해, 많은 비상태 멀티캐스트 프로토콜이 제안되었고, 이 프로토콜들은 데이터 패킷을 목적지로 보내기 위해 각각의 패킷 안에 목적지 주소 리스트를 적재한다. 그러나 데이터 패킷의 길이가 한정되어 있기 때문에, 목적지 주소 리스트 적재 정책은 프로토콜의 확장성을 제한한다. 이 논문에서는 이동 애드 혹 망을 위한 확장성이 있는 이동 기반 오버레이 멀티캐스트(SLAOM)라는 프로토콜을 제안한다. 이 프로토콜은 데이터 전달 구조 구성과 유지를 위한 주기적인 플러딩이 필요하지 않으며, 각각의 데이터 패킷에 목적지 주소 리스트를 적재할 필요도 없다. 시뮬레이션 결과는 우리의 프로토콜이 데이터 전송률과 제어 비용 면에서 다른 프로토콜들보다 뛰어남을 보여준다.

1. 서 론

지금까지 애드 혹 망을 위한 많은 멀티캐스트 프로토콜들이 제안되었다. 애드 혹 망의 동적인 특징들 때문에, 많은 멀티캐스트 프로토콜들이 데이터 전달 구조 구성과 유지, 멤버 갱신을 위한 막대한 양의 주기적인 플러딩 메시지로 인해 손해를 입었다. 특히 넓게 퍼져있는 애드 혹 망에서는 다수의 멀티캐스트 세션이 동시에 존재할 것이다. 이런 경우에 주기적인 플러딩 메시지는 네트워크 대역폭과 전지 자원을 심각하게 낭비하고, 프로토콜의 확장성과 효율성을 제한한다. 데이터 전달 구조 유지 부담을 줄이기 위해, 많은 비상태 멀티캐스트 프로토콜들이[1][2][3] 제안되었고, 이 프로토콜들은 데이터 패킷을 목적지로 보내기 위해 각각의 패킷 안에 목적지 주소 리스트를 적재한다. 그러나 데이터 패킷의 길이가 한정되어 있기 때문에, 목적지 주소 리스트 적재 정책은 프로토콜의 확장성을 제한한다.

정확한 (예를 들어, 정확성 오차가 몇 미터 이내인) GPS가 동기가 되어, 애드 혹 네트워크를 위한 많은 위치 인식 멀티캐스트 프로토콜이 제안되었다. 이런 위치 인식 멀티캐스트 프로토콜들은 멤버 노드들의 위치 정보를 이용하여 높은 효율성과 확장성을 제공한다. 이런 위치 인식 프로토콜들에서는 루트 검색과 유지를 위해 주기적인 플러딩 메시지를 사용하지 않으며, GPS가 노드의 이동을 추적한다.

LGT는 애드 혹 망을 위해 제안된 전형적인 위치 인식 멀티캐스트 프로토콜이다. 그러나 LGT의 세 가지 구축 메커니즘은 모든 멤버 노드들이 각각의 위치 정보를 알아야 한다. 따라서 망 전체를 범위로 하는 위치 갱신 메커니즘이 필요하다. 이런 전체적인 위치 갱신 메커니즘은 네트워크의 부하를 심각하게 증가시킨다. 게다가 LGT는 데이터 패킷을 보낼 때, 패킷 안에 들어있는 목적지 주소 리스트를 이용한다. 데이터 패킷의 크기는 제한이 있기 때문에, 일정 크기 이상의 목적지 주소 리스트는 하나의 데이터 패킷에 모두 넣을 수 없다. 이는 미래에 IPv4 대신 IPv6를 사용하게 될 경우 더욱 심각한 문제가 된다. 이런 제한 사항들 때문에 LGT는 확장성이 요구되는 멀티캐스트 그룹에 적합하지 않다.

앞에서 언급했듯이, 우리는 이 문제를 해결하기 위해 애드 혹 망을 위한 확장성이 있는 이동 기반 오버레이 멀티캐스트 프로토콜을 제안한다. SLAOM에서는 라우팅 구성과 유지 그리고 멤버 갱신을 위한 주기적인 플러딩 메시지가 필요하지 않다. 소스는 데이터 패킷을 보내기 위해 메모리에 기억된 모든 멤버 노드들의 위치 정보를 기반으로 클러스터 헤드로만 이루어진 Steiner 트리를 구성한다. 소스는 오버레이 Steiner 트리 구성 도중에 멤버 노드들 간의 거리에 따라 클러스터 헤드를 선정한다. Steiner 트리를 만들고 나면, 소스는 각각의 하위 트리를 Tree Distribution Packet(TDP)안에 캡슐화 한다. 이 TDP 패킷은 트리 분배에 사용한다. 트리 분배 후, 소스는 Steiner 트리를 따라 데이터 패킷을 전송하기 시작한다. 데이터 패킷을 수신한 클러스터 헤드는 IP-in-IP[4]

터널을 통해 하위 클러스터 헤드에게 데이터 패킷을 전달하는 동시에, 자신의 클러스터 멤버가 존재한다면 데이터 패킷을 브로드캐스트로 전송한다. LGT와는 달리 SLAOM은 데이터 패킷에 목적지 주소 리스트를 캡슐화하지 않는다.

이후 논문은 다음과 같이 구성된다: 2단원에서는 우리 프로토콜의 기초가 되는 가정을 언급한다. 3단원에서는 우리의 알고리즘을 트리 구성과 분배를 포함하여 자세히 설명한다. 성능 평가와 분석은 4단원에서 다루어진다. 5단원에서는 논문의 결론을 내린다.

2. 가 정

우리는 SLAOM 프로토콜에서 다음과 같은 가정을 만들었다: 소스 노드는 높은 계산 능력을 가졌다. Steiner 트리의 구성과 분할, 그리고 TDP 패킷의 적재로 인해 소스 노드에는 높은 계산 부하가 걸린다. 특히 확장성이 요구되는 멀티캐스트 그룹에는 더욱 그러하다. 각각의 노드는 GPS와 같은 위치 추적 서비스를 통해 자신의 위치 정보를 정확히 알 수 있다. 또한 각각의 노드는 자신의 전송 범위를 알고 있다. 우리의 클러스터링 알고리즘은 멤버 노드들의 전송 범위와 그들 사이의 거리를 기반으로 한다. 전송 범위보다 더 멀리 위치하는 두 노드는 서로에게 도달하기 위해 추가적인 네트워크 레벨 홉이 필요하다. 이 가정은 LGT에서 입증되었다.

3. SLAOM Overlay 구성과 분배

SLAOM은 Takahashi-Matsuyama의 학습식 방안에 의해 Steiner 트리를 구성한다. 1) 홉 수 대신 지리적인 거리를 사용한다. 2) 클러스터링 알고리즘은 Steiner 트리 구성에 따른다. 3) 클러스터 헤더만 트리 노드로 사용할 수 있다.

3.1 멤버 노드의 정보 수집과 갱신

소스 노드는 모든 멤버 노드의 정보를 멤버 노드 정보 테이블에 유지한다. SLAOM은 이 테이블에 따라 Steiner 트리를 구성한다. 하나의 멤버 노드가 그룹에 들어오면 소스에게 Membership Registration(MR) 메시지를 유니캐스트로 전송한다. 멤버 등록 메시지에는 멤버 노드의 아이디, 위치 정보, 전송 범위와 같은 정보가 들어있다. 그리고 멤버 노드는 주기적으로 자신의 위치를 점검한다. 만약 위치가 변화했다면, 소스가 멤버 노드 정보 테이블을 갱신할 수 있도록 소스에게 MR 메시지를 재전송한다. 소스는 MR 메시지를 받으면 테이블에서 이미 존재하는 멤버 노드인지 확인한다. 이미 존재한다면 MR 메시지로부터 받은 새로운 정보로 이를 갱신하고, 그렇지 않다면 새로운 멤버 노드로 테이블에 추가한다.

3.2 SLAOM 오버레이 Steiner 트리 구성

그림 1은 SLAOM의 Steiner 트리 구성 과정을 보여준다. 그룹 멤버가 아닌 모든 노드들은 그림 1에서 제외하였다. 최초로 Steiner 트리는 소스 노드인 S만을 포함하며, S는 자신을 클러스터 헤드로 설정한다. 각각의 클러스터 헤드는 오버레이 상에서 소스까지의 홉 거리를 나타내는 깊이 값을 가지고 있다. 각각의 클러스터 헤드는 또한 Pure Head Flag(PHF)를 가지고 있는데, 이는 "0" 이나 "1"을 값으로 가질 수 있다. "0"은 클러스터 헤드가 클러스터 멤버를 하나도 가지고 있지 않다는 뜻이며, "1"은 적어도 하나 이상의 클러스터 멤버를 가지고 있다는 뜻이다. 소스 S는 우선 깊이를 "0"으로 설정한다. 이후 소스는 자신으로부터 모든 멤버 노드들의 거리를 계산하여 소스의 신호 전송 범위의 반지름과 비교한다. 그림 1에서 멤버 노드 A와 B는 소스와의 거리가 소스의 신호 전송 범위의 반지름보다 작다. 이는 멤버 노드 A와 B가 소스 S의 전송 범위 안에 위치한다는 것을 의미하므로 소스 S는 PHF 플래그를 "1"로 설정하고 둘을 자신의 클러스터 멤버로 기록한다. A와 B처럼 클러스터 멤버로 기록되면, 현재의 Steiner 트리 구성에는 더 이상 참여할 수 없다.

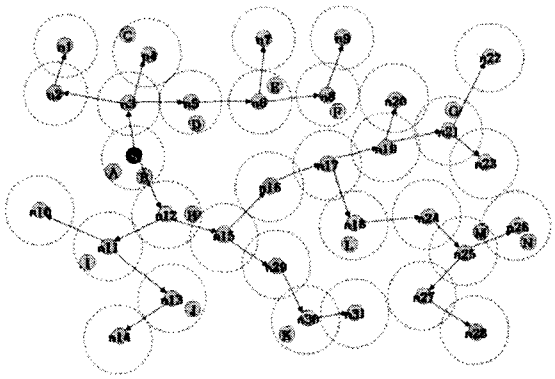

그림 1. SLAOM 오버레이 구성, 모든 그룹 멤버가 표시됐다.

소스는 남아있는 모든 멤버 노드들 중에 가장 가까운 노드를 선택하여 클러스터 헤드로 설정하고 Steiner 트리에 추가한다. 그림 1에서 멤버 노드 n3가 지리적으로 S로부터 가장 가깝기 때문에 소스 노드는 n3를 클러스터 헤드로 선정하고, Steiner 트리의 말단에 추가한다: S->n3. n3의 깊이는 "1"로 설정된다. 이는 Steiner 트리의 오버레이 상에서 n3로부터 S까지의 홉 수를 나타낸다. 이후 소스는 n3의 신호 전송 범위 안에 존재하는 멤버 노드들을 찾기 위해서 n3로부터 남아있는 모든 멤버 노드들의 거리를 계산하고 n3의 신호 전송 범위의 반지름과 비교한다. n3의 신호 전송 범위 안에 다른 멤버 노드들이 존재하지 않기 때문에 n3의 PHF 플래그를 "0"으로 설정한다.

다음 단계로, 트리에 포함되지 않은 멤버 노드들을 검토하고 지금까지 만들어진 트리에 가장 가까운 노드를 선택한다. 그림 1에서 S와 남아있는 노드들의 거리를 비교하는 동시에 n3와 남아있는 노드들의 거리를 비교한다. n3와 n4의 거리가 가장 가깝기 때문에 n4를 클러스

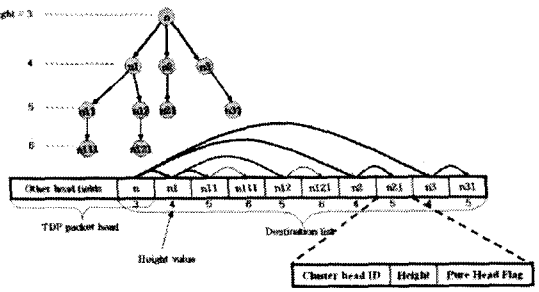

그림 2. TDP 패킷의 캡슐화 및 분배

터 헤드로 선정하고 Steiner 트리의 말단에 추가한다: n3->n4. Steiner 트리의 오버레이 상에서 n4로부터 S까지의 2홉이므로 n4의 깊이를 "2"로 설정한다. n4의 전송 거리 안에 멤버 노드 C가 있으므로 n4의 PHF 플래그를 "1"로 설정한다. 멤버 노드 C는 클러스터 멤버로 등록되며, 더 이상 Steiner 트리에 참여할 수 없다. 이 과정은 모든 노드들이 그림 1처럼 클러스터 헤드로 Steiner 트리에 포함되던지 단순히 클러스터 멤버로 선정될 때까지 반복된다. 이런 방식으로, 소스와 클러스터 헤드로만 구성된 Steiner트리가 만들어지고 소스가 이 트리의 루트가 된다.

3.3 SLAOM 하위 트리 캡슐화와 분배

Steiner 트리가 구성된 후, 소스는 각각의 하위 트리를 TDP 패킷에 캡슐화 하고, 각 TDP 패킷을 그 하위 트리의 루트에 유니캐스트로 전송한다.

그림 2는 하위 트리를 어떻게 TDP 패킷에 캡슐화 하는지 보여준다. 우선, 트리의 루트인 n을 TDP 패킷에 적재한다. 실제로 하나의 클러스터 헤드를 TDP 패킷에 적재한다는 것은 클러스터 헤드의 아이디와 깊이, PHF 모두를 적재한다는 것을 의미한다. 다음으로 하위의 클러스터 헤드들 중 임의의 하나를 TDP 패킷에 적재한다. 하위 트리 캡슐화는 다음의 규칙을 따른다: 하나의 클러스터 헤드를 TDP 패킷에 적재하면 형제 클러스터 헤드를 캡슐화 하기 전에 적재한 클러스터 헤드의 하위 트리를 모두 적재해야 한다. 이 규칙에 따라 TDP 패킷에 캡슐화된 트리는 트리 분배 과정에서 쉽게 재저장할 수 있다. 두 번째 단계에서, n1을 TDP 패킷에 적재한다. n1은 두 개의 하위 클러스터 헤드를 가지고 있기 때문에, 앞에서 언급한 규칙을 따르면 다음으로 적재할 클러스터 헤드는 n2가 아닌 n11이다. 그 다음으로 적재할 클러스터 헤드는 n12가 아닌 n111이다. n111은 하위 클러스터 헤드나 형제 클러스터 헤드가 없기 때문에, 다음으로 적재될 클러스터 헤드는 n12이고, 그 다음은 n121이다. 이제 n1의 하위 트리에 있는 모든 클러스터 헤드가 TDP 패킷에 캡슐화 되었다. 이제 루트 n의 또 다른 하위 클러스터 헤드인 n2를 TDP 패킷에 캡슐화 한다. 이 과정은 제시된 트리의 모든 클러스터 헤드가 TDP 패킷에 적재될 때까지 반복된다. 그림 2의 TDP 패킷은 클러스터 헤드의 캡슐화 순서를 보여준다. 소스 노드의 모든 다른 하위 트리들도 같은 방법으로 TDP 패킷에 캡슐화 된다. 앞에 언급한 Steiner 트리 구성과 TDP 패킷 캡슐화는 모두 소스 노드의 메모리에서 이루어진다. 모든 멤버 노드들은 Steiner 트리의 어떤 정보도 알지 못한다.

TDP 패킷을 캡슐화한 후, 소스는 각각 상응하는 하위 트리의 루트에게 TDP 패킷을 유니캐스트로 전송한다. 그림 2에서 이 과정을 적용하면, 소스는 루트 n에게 TDP 패킷을 유니캐스트로 전송한다. 루트 n은 TDP 패킷을 받은 후, 자신에게 해당하는 깊이와 PHF를 저장하고 TDP 패킷의 목적지 주소 리스트에서 자신의 아이디, 깊이, PHF를 삭제한다. 그런 후 리스트에서 자신의 깊이보다 1만큼 더 큰 깊이를 가지고 있는 클러스터 헤드를 찾는다. 그림 2의 리스트에서 클러스터 헤드 n1, n2, n3가 이에 해당한다. 이것은 루트 n의 입장에서 하위 클러스터 헤드가 n1, n2, n3 이렇게 3개라는 것을 인식한다는 것을 의미한다. 루트 n은 3개의 클러스터 헤드를 자신의 하위 클러스터 헤드로 저장한다. 처음 받았던 TDP 패킷은 이제 하위 클러스터 헤드를 기준으로 세 개의 부분으로 나눠진다. 우리의 TDP 패킷 캡슐화 규칙에 따르면, 리스트에서 n1과 n2 사이의 부분은 모두 n1의 하위 트리에 소속된 클러스터 헤드이다. 마찬가지로 n2와 n3 사이의 부분은 모두 n2의 하위 트리에 소속된 클러스터 헤드이며, n3부터 끝까지의 부분은 모두 n3의 하위 트리에 소속된 클러스터 헤드이다. 루트 n은 n1을 위해 목적지가 n1인 TDP 패킷을 만들고, 이 패킷 안에 리스트 <n11, n111, n12, n121>을 캡슐화 한다. 그리고 이 TDP 패킷을 n1에게 유니캐스트로 전송한다. n1과 마찬가지로, 루트 n은 n2와 n3를 위해 리스트가 각각 <n21>, <n31>인 TDP 패킷을 만들어 유티캐스트로 전송한다. 이 과정을 리스트의 모든 클러스터 헤드에 TDP 패킷이 도달할 때까지 반복한다. 이것이 Steiner 트리 분배 방안이다. 이런 방식으로 최초 소스만이 알고 있던 Steiner 트리를 트리의 모든 클러스터 헤드에게 전파할 수 있다. Steiner 트리의 분배 후, 최초의 Steiner 트리는 소스의 메모리에서 삭제되지만, 모든 멤버 노드의 정보는 멤버 노드 정보 테이블에 계속 남는다.

3.4 SLAOM에서의 데이터 전달

Steiner 트리 분배가 끝나면, 소스는 오버레이 Steiner 트리를 따라서 모든 그룹 멤버들에게 데이터를 전달하기 시작한다. 클러스터 헤드는 데이터 패킷을 받으면, 첫째로 PHF 플래그를 검사한다. 만약 "1"이면 해당 클러스터 헤드는 적어도 하나 이상의 클러스터 멤버를 가지고 있다는 뜻이므로, 받은 데이터 패킷을 클러스터 멤버들에게 브로드캐스트로 전송한다. 둘째로 모든 하위 클러스터 헤드에게 IP-in-IP 터널을 통해서 받은 데이터 패킷을 복사, 전송한다.

4. 성능 분석

LGT와 SLAOM 모두 데이터 전달을 위해서 멤버 노드

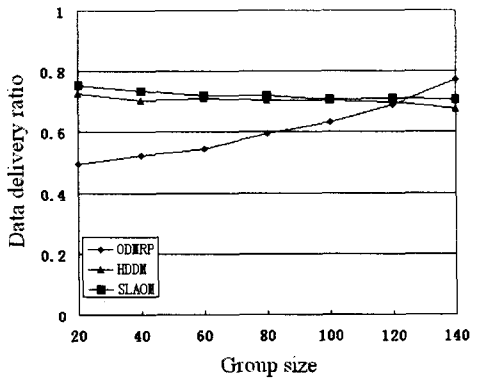

그림 3. 그룹 크기에 따른 ODMRP, HDDM, SLAOM의 데이터 전송률

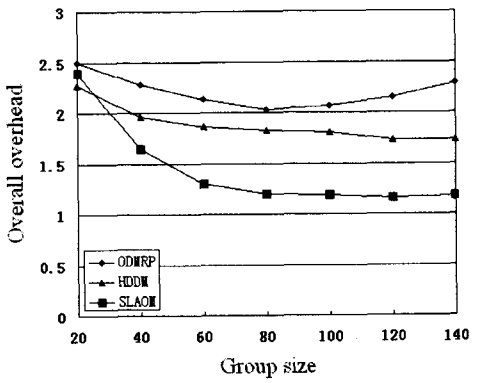

그림 4. 그룹 크기에 따른 ODMRP, HDDM, SLAOM의 전체적인 부하

의 위치 정보를 기반으로 오버레이 Steiner 트리를 구성한다. 이 단원에서는 종합적인 부하의 관점에서 이 두 프로토콜을 평가한다.

4.1 시뮬레이션 환경

시뮬레이션은 2000*2000m^{2}의 공간에서 150개의 이동 호스트들이 무작위로 위치하도록 설계했다. 각각의 노드는 전송 범위가 200m이며, 채널 용량이 2Mbit/s이다. 이동 모델은 정지 시간이 변화하는 임의 중간지점 모델을 사용한다. 무선 전송 모델은 송수신 겸용 기반 모델을 사용했다. 데이터는 상수 비트율로 발생한다(1 bits/second). 마지막으로, MAC 계층 프로토콜로 IEEE 802.11을 사용했다. 속도 범위는 0m/s에서 2m/s로 설정했으며, 시뮬레이션 시간은 600s이다.

4.2 ODMEP, HDDM과 비교 측정

ODMRP는 데이터 전달을 위해 그물형 구조를 구성하고, HDDM은 데이터 전달을 위해 계층적 구조를 구성한다. 둘 모두 확장성 있는 멀티캐스트 그룹에 적용할 수 있다. 우리는 멀티캐스트 그룹의 크기를 20에서부터 150까지로 변경하였다.

그림 3은 패킷 전송률을 보여준다. 데이터 전송률은 수신자에게 전달된 데이터 패킷의 비율로 정의된다. 그룹의 크기가 커질수록 ODMRP는 더 많은 패킷이 전달된다. 그 이유는 그물형 구조의 특성상 크기가 커질수록 더 많은 중복 진로가 생겨나 신뢰성이 높아지기 때문이다. HDDM 프로토콜은 근소하게 줄어드는 추세로 안정된 전송률을 기록한다. 계층적 트리는 보다 큰 그룹에서는 신뢰성이 낮아지게 되는데, HDDM은 그룹의 크기에 상관없이 항상 계층적 트리를 전달 구조로 사용한다. 반면 SLAOM에서는 그룹 크기가 커질수록 더 많은 멤버 노드들이 가까이에 위치할 확률도 함께 증가한다. clustrering 알고리즘을 사용함으로써, 그룹의 크기가 증가하더라도 트리의 링크수가 그에 비례하여 증가하지 않는다. 따라서 그룹 크기의 증가가 데이터 전송률에 미치는 영향이 작다.

그림 4는 세 프로토콜의 전체적인 부하를 나타낸다. 여기서 우리는 SLAOM이 다른 두 개의 프로토콜보다 전체적인 부하 면에서 매우 효과적이라는 것을 알 수 있다. SLAOM에서는 각각의 데이터 패킷에 목적지 주소 리스트가 들어있지 않다. 그룹의 크기가 커지면 단지 TDP 패킷의 크기만 증가한다. TDP 패킷의 크기 증가는 전체적인 부하에 많은 영향을 주지 않는다. 따라서 "각 멤버의 부하"는 그룹 크기가 증가하면서 두드러지게 감소한다. HDDM에서는 각각의 패킷에 목적지 주소 리스트(멤버 주소와 하위 루트들의 주소)가 캡슐화되어 있기 때문에 그룹의 크기가 증가함에 따라 패킷의 크기가 같이 증가한다. 목적지 주소 리스트가 각각의 데이터 패킷에 캡슐화 되어 있기 때문에 그룹의 크기가 커짐에 따라 각 멤버의 부하가 줄어드는 폭이 매우 작다. ODMRP의 곡선을 보면 처음 그룹의 크기가 커질 때 전체적인 부하가 감소한다. 제어 패킷의 수가 증가하더라도, 전달되는 패킷의 수가 더 빨리 증가하기 때문이다. 그러나 그룹의 크기가 80 이상이 될 때 곡선은 다시 증가한다. 그 이유는 수신자들이 보내는 JOIN_REPLY 패킷이 더 빈번하게 충돌하면서 JOIN_REPLY 패킷을 재전송 하는 횟수가 급격하게 증가하게 되기 때문이다.

5. 결 론

이 논문에서 우리는 SLAOM이라는 이동 애드 혹 망에서 확장성 있는 오버레이 멀티캐스트 프로토콜을 제안했다. SLAOM에서 소스는 멤버 노드들의 위치 정보를 바탕으로 오버레이 Steiner 트리를 구성한다. 트리를 구성한 후, 소스는 TDP 패킷 안에 각각의 하위 트리를 캡슐화 하고, 이 패킷을 통해 모든 멤버 노드들에게 Steiner 트리를 분배한다. 따라서 각각의 패킷에 목적지 주소 리스트를 적재하지 않아도 된다. 시뮬레이션 결과는 작은 망에서 뿐만 아니라 확장성을 지원하는 멀티캐스트 그룹에서도 높은 성능을 가진다는 것을 보여준다.

참고 문헌

[1] K. Chen and K. Nahrstedt, "Effective location-guided tree construction algorithms for small group multicast in manet", in Proc. of the IEEE Infocom, 2002, pp. 1180-1189.

[2] C. Gui and P. Mohapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks," Wireless Communications and Networking Conference (WCNC), March. 2003, pp.1118-1123.

[3] L. Ji and MS Corson, "Differential destination multicast—A MANET multicast routing protocol for small groups," in Proc. of the IEEE Infocom2001, pp.1192-1202.

[4] C.E. Perkins. "IP Encapsulation within IP". RFC 2003, Internet Engineering Task Force, October 1996.

[5] S.-J. Lee and M. Gerla and C.-C. Chiang, "On-Demand Multicast Routing Protocol", in Proc. of IEEE WCNC'99, New Orleans, Sept. 1999, pp. 1298-1304

[6] Gui, C. Mohapatra, P. "Scalable multicasting in mobile ad hoc networks", in Proc. of the IEEE Infocom2001, Volume 3, 2004 Page(s): 2119 - 2129