애드-혹 네트워크에서의 확장성 있는 다중점 대 다중점 라우팅 프로토콜

(A Scalable Multipoint-to-Multipoint Routing Protocol in Ad-Hoc Networks)

강 현 정 † 이 미 정 ††

(Hyunjeong Kang) (Meejeong Lee)

요 약 기존의 애드-혹 네트워크를 위한 멀티캐스트 프로토콜들에서는 송신원 수가 많은 경우의 프로토콜 효율성을 고려하지 않아, 송신원 수가 많아지는 경우 프로토콜 오버헤드가 지나치게 커지거나, 데이타 전달율이 저하되는 결과를 가져올 수 있다. 이에 본 논문에서는 멀티캐스트 그룹의 송신원 수에 대한 확장성을 고려한 애드-혹 네트워크를 위한 멀티캐스트 라우팅 프로토콜을 제안한다. 제안하는 프로토콜은 송신원 중 일정 비율을 코어 송신원으로 선택하고, 선출된 코어 송신원을 루트로 하여 각 코어 송신원으로부터 멀티캐스트 그룹의 모든 수신원에 이르는 코어 송신원별 트리를 구성한다. 이렇게 구성된 코어 송신원별 트리의 합집합으로 데이타 전달 메쉬를 형성하고, 일반 송신원들은 가장 가까운 곳에 위치한 코어 송신원을 선택하여 해당 코어 송신원을 통해 데이타 패킷을 전송하게 된다. 제안하는 프로토콜이 효율적으로 동작하기 위해서는 적절한 수의 코어 송신원을 선출하는 것이 중요하다. 너무 많은 수의 코어 송신원을 선출하게 되면, 데이타 전달 경로를 유지하기 위해 코어 송신원이 주기적으로 플러딩하는 제어 메시지 오버헤드나 불필요한 중복 데이타 패킷 오버헤드가 과다하게 된다. 반면에 너무 적은 수의 코어 송신원은 호스트의 이동성에 대해서 안정적인 경로를 제공하지 못하고 전달 트리 상에 과부하가 발생할 수 있어 데이타 전달율을 저하시키는 결과를 가져온다. 제안하는 프로토콜은 코어 송신원이 주기적으로 플러딩하는 제어 메시지를 통해 데이타 전달 메쉬를 최적으로 재구성하고, 주기적인 최적 재구성 기간 사이에는 지속적인 메쉬의 연결 유지를 위해 국부적 메쉬 재구성을 수행한다. 시뮬레이션을 통하여 기존에 제안된 프로토콜들과 성능을 비교한 결과, 제안하는 프로토콜이 멀티캐스트 그룹의 송신원 수가 많은 경우에 데이타 전달율 및 오버헤드 측면에서 보다 효율적인 멀티캐스트 통신을 제공함을 알 수 있었다.

키워드 : 애드-혹 네트워크, 멀티캐스트, 메쉬 구조, 코어 송신원, 국부적 메쉬 재구성

Abstract Most of the existing multicast routing protocols for ad-hoc networks do not take into account the efficiency of the protocol for the cases when there are large number of sources in the multicast group, resulting in either large overhead or poor data delivery ratio when the number of sources is large. In this paper, we propose a multicast routing protocol for ad-hoc networks, which particularly considers the scalability of the protocol in terms of the number of sources in the multicast groups. The proposed protocol designates a set of sources as the core sources. Each core source is a root of each tree that reaches all the destinations of the multicast group. The union of these trees constitutes the data delivery mesh, and each of the non-core sources finds the nearest core source in order to delegate its data delivery. For the efficient operation of the proposed protocol, it is important to have an appropriate number of core sources. Having too many of the core sources incurs excessive control and data packet overhead, whereas having too little of them results in a vulnerable and overloaded data delivery mesh. The data delivery mesh is optimally reconfigured through the periodic

† 비 회 원 : 이화여자대학교 컴퓨터학과
semper@hanmir.com
†† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
lmj@mm.ewha.ac.kr
논문접수 : 2002년 1월 7일
심사완료 : 2003년 1월 21일

control message flooding from the core sources, whereas the connectivity of the mesh is maintained by a persistent local mesh recovery mechanism. The simulation results show that the proposed protocol achieves an efficient multicast communication with high data delivery ratio and low communication overhead compared with the other existing multicast routing protocols when there are multiple sources in the multicast group.

Key words : ad-hoc networks, multicast, mesh structure, core source, local mesh recovery

1. 서 론

애드-혹 네트워크는 고정된 네트워크 기반 없이 이동하는 호스트들로 이루어진 다중 홉 무선 네트워크이다. 애드-혹 네트워크에서 이동하는 호스트들은 고정 네트워크에서의 라우터처럼 데이타를 포워딩하는 기능을 담당한다. 애드-혹 네트워크는 호스트들의 이동성으로 인해 네트워크 구조가 예측할 수 없이 변화하며, 애드-혹 네트워크를 구성하는 각 호스트들은 제한된 데이타 처리 능력과 저장 공간을 갖고 있다. 이러한 애드-혹 네트워크의 대표적인 응용으로는 기반 구조가 없는 재난/전투 현장 혹은 컨퍼런스 등에서의 통신을 들 수 있다. 그런데, 이러한 응용들은 대부분 다중점 대 다중점 통신을 요구하기 때문에 애드-혹 네트워크에서의 효율적인 멀티캐스트 지원이 요구된다. 이에 따라, 최근 애드-혹 네트워크의 특성을 고려한 멀티캐스트 라우팅 프로토콜에 대한 연구가 이루어지고 있다[1]. 그런데 이들 기존의 연구에서 제안하는 멀티캐스트 프로토콜들에서는 다중 송신원 멀티캐스트 그룹의 경우에 송신원 수가 증가함으로써 발생할 수 있는 확장성 문제에 대해서는 고려된 바가 별로 없다. 이에, 본 논문에서는 송신원이 여러 개인 멀티캐스트 그룹을 효율적으로 지원할 수 있는 확장성 있는 멀티캐스트 라우팅 프로토콜을 제안한다.

지금까지 제안된 애드-혹 네트워크를 위한 멀티캐스트 프로토콜들은 데이타 전달 구조의 형태에 따라 크게 트리 기반 방식과 메쉬 기반 방식으로 나눌 수 있다. 트리 기반 프로토콜에서는 데이타 전달 구조로서 하나의 특정 노드가 루트가 되는 단일 트리를 사용하며, 이 데이타 전달 트리 상의 각 노드는 부모 노드로부터 온 데이타만을 포워딩한다. 반면에, 메쉬 기반 프로토콜은 일반적으로 하나 이상의 트리들의 집합으로 이루어진 데이타 전달 구조를 형성하고 데이타 전달 구조상의 각 노드는 해당 멀티캐스트 그룹에 속하는 데이타는 모두 전달한다.

대표적인 트리기반 프로토콜로는 AMRIS(Ad hoc Multicast Routing protocol utilizing Increasing id-numberS), AMRoute(Adhoc Multicast Routing), MAODV(Multicast operation of the Ad-hoc On-demand Distance Vector routing protocol) 등을 들 수 있는데, 이 프로토콜들은 각각의 멀티캐스트 그룹의 멤버에게 이르는 단 하나의 경로를 제공하기 때문에 호스트의 이동성이 커지게 되면 이에 대응하기 위한 오버헤드가 매우 커지거나 경로 단절로 인한 패킷 손실이 커지게 된다[2,3,4,5,6]. 뿐만 아니라, 멀티캐스트 그룹의 송신원 수가 증가하는 경우에는, 모든 송신원이 내보내는 데이타가 하나의 트리를 통해서 전달되기 때문에 데이타 전달 트리 상에 혼잡이 발생할 수 있다.

대표적인 메쉬 기반 프로토콜로는 ODMRP(On-Demand Multicast Routing Protocol)나 CAMP (Core-Assisted Mesh Protocol) 등을 들 수 있는데, ODMRP나 CAMP는 모두 각 송신원으로부터 모든 수신원에 이르는 최단 경로상의 노드들로 데이타 전달 메쉬를 구성한다[7,8,9,10]. 즉, 멀티캐스트 그룹의 각 송신원을 루트로 하여 모든 수신원에 이르는 송신원별 트리의 합집합으로 데이타 전달 메쉬를 구성하게 되는데, 이로 인해 멀티캐스트 그룹의 송신원 수가 증가할수록 데이타 전달 트리의 수가 증가하고 이를 유지하기 위한 오버헤드가 커지게 된다. 또한, 송신원별 트리 수의 증가로 데이타 전달 메쉬가 필요 이상으로 지나치게 두터워지면 중복된 데이타 패킷 전달로 인한 오버헤드도 커지게 된다. 특히, CAMP는 송신원별 트리 외에도, 멀티캐스트 그룹 멤버들이 처음 멀티캐스트 세션에 참여할 때 데이타 전달 메쉬에 연결되기 위해 알려진 코어 노드 중 하나에 연결하게 되는데, 이들 코어 노드들 간에는 완전 메쉬를 형성하기 때문에 코어 메쉬 주변에 패킷들이 많이 집중하는 현상이 발생할 수 있다. 송신원 수가 증가하면 코어 메쉬에서의 혼잡 현상은 더욱 심해지게 된다. [11]은 ODMRP의 데이타 전달 메쉬 유지를 위한 제어 메시지 플러딩 오버헤드를 줄이기 위한 방안을 제시하였으나, ODMRP와 마찬가지로 송신원별 트리를 유지하기 때문에 여전히 위에서 설명한 송신원 수 증가에 대한 프로토콜 확장성 문제를 겪을 수 있다.

Ozaki 등이 제안한 '대역폭 측면에서 효율적인 멀티

캐스트 라우팅 프로토콜'은 트리 같은 형태의 데이타 전달 구조를 사용하지만, 다른 트리 기반 방식들과는 달리 트리의 중심이 되는 루트 노드에 대한 개념이 없으며 트리를 최적화하지도 않는다[12]. 이 방식은 기존의 트리 기반 방식에 비해서는 트리 유지 오버헤드가 적다는 장점이 있으나, 데이타 전달 구조 형태가 단일 트리이기 때문에 송신원 수 증가에 따른 트리 상에서의 데이타 양의 증가에 대한 확장성 문제는 다른 트리 기반 방식들과 동일하다.

본 논문에서는 송신원 수가 많은 상황에서 효율적인 멀티캐스트 통신을 제공할 수 있는 프로토콜인 SMMRP(Scalable Multi-source Multicast Routing Protocol)를 제안한다. SMMRP는 하나의 노드가 루트가 되는 단일 트리 구조나 송신원별 트리의 합집합으로 이루어진 데이타 전달 메쉬 구조가 아니라 송신원별 트리의 부분 집합으로 이루어진 데이타 전달 메쉬 구조를 사용한다. 즉, SMMRP는 송신원들 중 일부를 코어 송신원으로 선택하고 이들 코어 송신원들이 루트가 되는 트리들의 합집합으로 데이타 전달 메쉬를 형성한다. 송신원 수에 대한 코어 송신원 수의 비율은 SMMRP의 성능에 영향을 주는 중요한 변수이다. 적절한 수의 코어 송신원을 선택함으로써 데이타 전달 메쉬를 유지하는 오버헤드 절약과 노드의 이동성에 대한 데이타 전달 구조의 안정성 사이에 균형을 취할 수 있어야 한다. SMMRP에서는 애드-혹 네트워크 도메인에 알려진 서버가 있다고 가정하고 이 서버가 모든 멀티캐스트 그룹의 송신원들에 대한 정보를 모아, 이들 중 적절한 수를 코어 송신원으로 선택하도록 한다. 그리고 코어 송신원이 아닌 일반 송신원은 자신으로부터 가장 가까운 곳에 위치한 코어 송신원을 선택하여 그 코어 송신원에게 패킷 전달을 의뢰하도록 한다. 기존의 메쉬 기반 방식 가운데 CAMP도 전달 메쉬를 구성하는데 코어를 사용한다. 그러나 CAMP에서는 그룹 멤버들이 참여 요청 메시지를 코어를 향해 유니캐스트하여 전달 메쉬를 생성하는데 발생하는 제어 메시지 양을 감소시키기 위한 목적으로 코어를 사용하는데 반해, SMMRP에서는 송신원별 트리가 지나치게 많아져서 패킷이 중복해서 네트워크 상에 발생하는 것을 제한하기 위한 목적으로 코어를 도입하였다. 따라서 두 스킴의 코어 사용 목적은 근본적으로 차이가 있다.

SMMRP는 코어 송신원이 플러딩하는 제어 메시지를 통해 데이타 전달 메쉬를 주기적으로 재구성하고, 지속적인 국부적 메쉬 재구성에 의하여 주기적인 메쉬 재구성을 보완한다. SMMRP는 ODMRP나 CAMP와 같은 메쉬 기반 방식에 비하여 메쉬를 구성하는 트리 수가 적기 때문에, 국부적인 메쉬 재구성을 위한 오버헤드도 더 적은 수의 애드-혹 네트워크 노드에 국한될 수 있어 더 효율적으로 국부적인 메쉬 재구성이 이루어 질 수 있다. CAMP에서 송·수신원은 멀티캐스트 그룹에 참여하기 위한 참여 요청 메시지를 코어를 향하여 유니캐스트하므로 CAMP에서 코어는 메쉬를 생성하는데 발생하는 제어 메시지를 제한하는데 사용되지만, SMMRP에서 코어 송신원은 일반 송신원이 발생하는 패킷을 수신원에게 전달하는데 사용되므로 패킷이 중복해서 네트워크 상에 발생하는 것을 제한할 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 본 논문에서 제안하는 프로토콜에 대해서 상세히 설명한다. 3장에서는 제안하는 프로토콜과 기존에 제안된 프로토콜들의 성능을 평가하기 위한 시뮬레이션 모델을 살펴보고 시뮬레이션 결과를 분석한다. 마지막으로 4장에서는 본 논문의 결론을 제시한다.

2. SMMRP

이 장에서는 SMMRP의 상세한 동작에 대해 설명한다. 서버가 각 멀티캐스트 그룹에 대한 송신원 목록을 유지하도록 하기 위해 각 송신원은 어떤 멀티캐스트 그룹에 참여하거나 탈퇴하는 경우 서버에게 이를 알려야 한다. 서버는 멀티캐스트 그룹의 멤버일 필요는 없으며, 애드-혹 네트워크 상의 모든 호스트들에게 알려져야 한다. 애드-혹 네트워크를 인터넷에 연결하는 엑세스 포인트의 역할을 하는 라우터들은 고정적이고 여러 가지 이유로 애드-혹 네트워크 내의 다른 라우터들에게 그 존재가 알려져 있기 때문에 이러한 라우터가 서버로서 동작한다고 가정한다. 서버는 각 멀티캐스트 그룹의 송신원 정보를 수집하고, 이들 중에서 코어 송신원을 선출, 다른 멤버들에게 이를 알리는 역할을 담당한다. 각 멀티캐스트 그룹에 대해 서버는 송신원들 중에서 적절한 수의 코어 송신원을 선출한다. 각 멀티캐스트 그룹의 송신원들은 자신으로부터 가장 가까운 코어 송신원을 결정하여, 그 코어 송신원에게 데이타 패킷을 전송하고, 코어 송신원은 자신에게 전달된 데이타 패킷을 수신원에게 전송한다.

그림 1은 SMMRP의 메쉬 구조를 보여준다. SMMRP의 메쉬 구조는 송신원-코어 메쉬와 코어-수신원 메쉬로 이루어진다. 송신원-코어 메쉬는 각 송신원으로부터 가장 가까운 코어 송신원에 이르는 트리들로 이루어진다. 코어-수신원 메쉬는 각 코어 송신원을 루트로 하여 해당 멀티캐스트 그룹의 모든 수신원들에 이르는 트리들

로 구성된다. 이들 두 메쉬 상에 있는 중간 노드들을 메쉬 노드라 부르기로 한다. 어떤 멀티캐스트 그룹의 메쉬 노드로 동작하는 노드들은 그 멀티캐스트 그룹에 속한 데이타 패킷을 받아서 전달하는 역할을 담당한다. 송신원-코어 메쉬와 코어-수신원 메쉬는 주기적으로 재구성되고, 각각 송신원으로부터 코어, 코어로부터 수신원들에 이르는 최단 경로를 유지하게 된다. 주기적인 메쉬 재구성 기간 사이에 각 메쉬 노드는 상위 메쉬 노드로 가는 경로를 감시하고, 상위 메쉬 노드로 가는 경로가 끊어진 경우에는 국부적인 제어 메시지 플러딩을 통해 새로운 상위 메쉬 노드를 찾아서 데이타 전달 메쉬를 유지한다. 멀티캐스트 데이타 전달은 각 메쉬 노드와 코어 송신원이 자신이 담당하는 멀티캐스트 그룹의 데이타를 받았을 때 이를 플러딩함으로써 이루어진다. SMMRP는 유니캐스트 라우팅 프로토콜을 사용하지 않고 단독적으로 사용될 수 있다.

2.1절에서 주기적인 메쉬 재구성에 대하여 설명하고, 이에 이어 2.2절에서는 지속적인 국부적 메쉬 재구성에 대하여 설명한다.

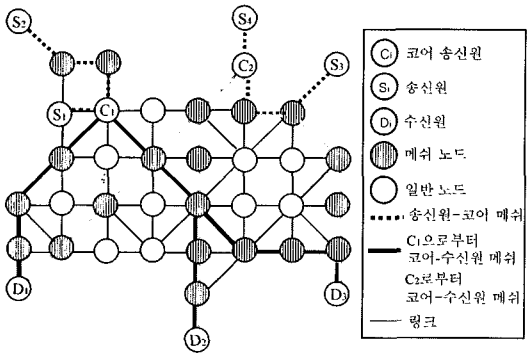

그림 1 SMMRP의 메쉬 구조

2.1 주기적인 메쉬 재구성

그림 2와 3은 각각 주기적인 메쉬 재구성과 관련이 있는 SMMRP의 자료구조와 제어 메시지들이다. 먼저, 데이타 전달 메쉬를 구성하는 모든 메쉬 멤버들은 그림 2(a)와 같이 메쉬 멤버 테이블을 유지한다. 메쉬 멤버 테이블은 해당 노드가 어떤 멀티캐스트 그룹의 메쉬 노드로서 언제까지 동작해야 하는가를 표시하기 위한 테이블로서, 멀티캐스트 그룹 ID, 해당 멀티캐스트 그룹의 메쉬 노드인지 여부를 표시하는 멤버 플래그, 해당 멀티캐스트 그룹의 메쉬 노드로서 언제까지 동작해야 하는지를 표시하는 타이머를 둔다. 코어 송신원들은 어느 멀티캐스트 그룹의 코어 송신원으로 동작하는가에 대한 정보를 표시하기 위해 그림 2(b)와 같은 코어 테이블을 둔다. 코어 테이블에는 멀티캐스트 그룹 ID, 해당 멀티캐스트 그룹의 코어 송신원인지 여부를 표시하는 코어 플래그, 그리고 해당 그룹에서 탈퇴할 경우에 코어 송신원 역할 만기 시간을 설정하는 타이머를 둔다. 서버는 각 멀티캐스트 그룹에 대하여 그 멀티캐스트 그룹에 속하는 송신원의 목록을 유지하는 송신원 목록 테이블(그림 2(c))과 코어 송신원으로 선출된 노드들의 목록을 유지하는 코어 송신원 목록 테이블(그림 2(d))을 둔다.

서버는 자신의 존재와 자신에게 이르는 라우팅 정보를 애드-혹 네트워크 전체에 알리기 위해 주기적으로 SV 메시지(그림 3(a))를 플러딩한다. SV 메시지를 받

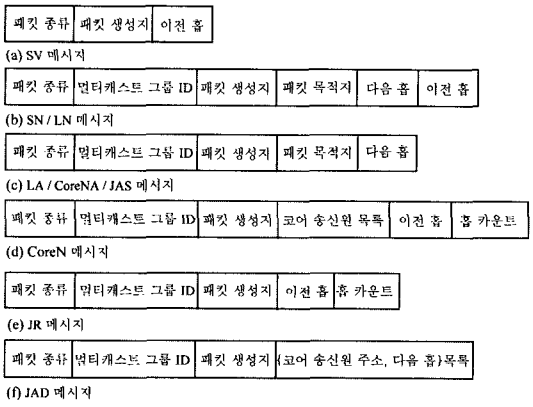

그림 2 주기적인 메쉬 재구성에 관련된 SMMRP의 자료구조

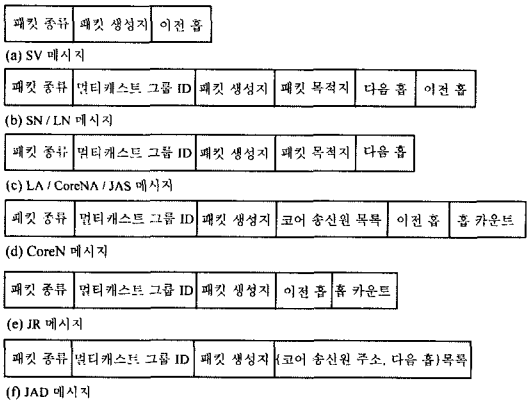

그림 3 주기적인 메쉬 재구성에 관련된 SMMRP의 제어메시지

은 각 노드는 자신의 라우팅 테이블에 서버의 주소와 SV 메시지를 전달해준 이전 홉 노드를 기록한다. 송신원은 멀티캐스트 그룹에 참여(탈퇴)할 때 서버에게 SN(LN) 메시지(그림 3(b))를 전송한다. SN(LN) 메시지를 서버에게 전달하는 중간 노드들은 SN(LN) 메시지를 생성한 송신원으로 가는 라우팅 정보를 기록한다. 서버가 SN(LN) 메시지를 받으면, 서버는 송신원 목록 테이블에서 SN(LN) 메시지에 명시된 멀티캐스트 그룹에 해당하는 엔트리를 찾아 SN(LN) 메시지를 생성한 송신원을 추가(제거)한다. 그리고 LN 메시지에 대한 응답으로 해당 송신원에게 LA 메시지(그림 3(d))를 보낸다. LA 메시지는 LN 메시지가 포워딩될 때 기록된 라우팅 정보를 이용해서 해당 송신원에게 전달된다. LN 메시지를 비롯하여 응답을 요구하는 메시지에 대해서는 모두 그 메시지에 대한 응답을 받을 때까지 최대 한도 횟수까지 재전송을 시도한다.

송신원 중 일정한 비율을 코어 송신원으로 선출하기 때문에 송신원의 수가 변함에 따라 코어 송신원의 수도 변경해야 할 경우가 있다. 새로운 송신원을 코어 송신원으로 선출하거나 기존의 코어 송신원을 제거해야 하는 경우, 서버는 CoreN 메시지(그림 3(d))를 만들어서 플러딩한다. CoreN 메시지를 받은 해당 송신원, 즉 새로이 코어로 선출되거나 코어에서 제거될 송신원은 코어 테이블의 해당 멀티캐스트 그룹 엔트리를 수정하고, CoreN 메시지에 대한 응답으로 CoreNA 메시지(그림 3(c))를 서버에게 전송한다. CoreNA를 받은 서버는 코어 송신원 목록 테이블에서 해당 멀티캐스트 그룹 엔트리를 수정한다. 나머지 송신원들은 서버가 만들어서 보낸 CoreN 메시지를 받으면 지금까지 자신들과 연결되어 있던 코어 송신원을 변경해야 하는지를 점검하고, 필요하다면 새로운 코어 송신원을 선택하여 그 코어 송신원으로의 송신원-코어 메쉬를 새롭게 구성한다.

코어 송신원은 주기적으로 자신을 알리고 자신에게 이르는 라우팅 정보를 갱신하기 위해 JR 메시지(그림 3(e))를 플러딩한다. JR 메시지가 플러딩될 때, 각 노드는 해당 JR 메시지를 만든 코어 송신원으로 가는 라우팅 정보를 갱신한다. 멀티캐스트 그룹의 송신원들은 JR 메시지를 통해 멀티캐스트 그룹의 코어 송신원 정보를 모으고, JR 메시지의 홉 카운트 필드를 통해 JR 메시지를 만든 코어 송신원으로부터의 거리 정보를 파악한다. 송신원들은 주기적으로 JR 메시지를 보낸 코어 송신원들 가운데 자신과 가장 가까운 코어 송신원을 선택하고 그 코어 송신원에게 JAS 메시지(그림 3(c))를 만들어 보낸다. 이 때 JAS 메시지는 JR 메시지가 플러딩될 때 기록되었던 라우팅 정보를 이용하여 전송된다. JAS 메시지를 받은 노드는 자신이 JAS 메시지에 기록된 다음 홉 노드에 해당하는지를 체크하고, 만약 그렇다면 메쉬 멤버 테이블에서 해당 멀티캐스트 그룹에 대한 엔트리를 수정하고 그 멀티캐스트 그룹의 메쉬 노드로서 동작하게 된다. 또한, JAS 메시지의 다음 홉 필드를 자신으로부터 해당 코어 송신원에 이르기 위한 다음 홉으로 변경하여 이를 다시 전달한다. 그림 4는 송신원-코어 메쉬가 구성되는 과정을 보여준다.

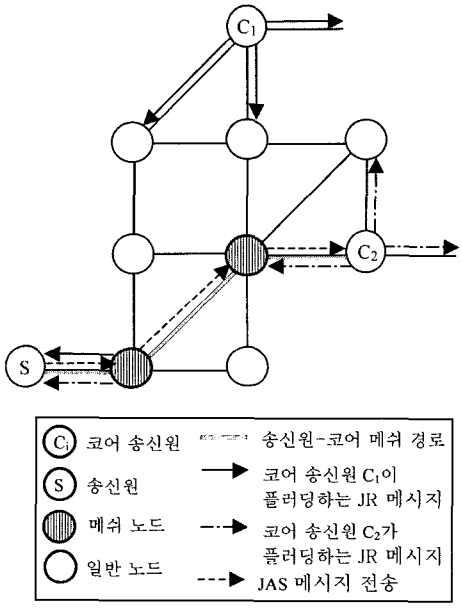

그림 4 송신원-코어 메쉬 형성

송신원과 유사하게, 멀티캐스트 그룹의 수신원들도 코어 송신원이 플러딩한 JR 메시지를 통해서 코어 송신원 정보를 파악한다. 그런데, 송신원과는 달리 수신원들은 모든 코어 송신원에 대하여 주기적으로 메쉬를 재구성한다. 각 수신원과 코어 송신원들 사이에 코어-수신원 메쉬를 구성하는 방법은 ODMRP에서 수신원이 각 송신원에 이르는 메쉬를 구성하는 것과 유사하게 진행된다[7,8]. 단, ODMRP에서는 각 수신원이 모든 송신원으로의 메쉬를 재구성한 것에 반해 제안하는 스킴에서는 코어 송신원들에 이르는 메쉬만을 재구성한다. 주기적인 코어-수신원 메쉬 재구성을 위하여 수신원들은 주기적으로 모든 코어 송신원을 향하여 JAD 메시지(그림 3(f))를 보낸다. JAD 메시지를 받은 노드는 자신이 JAD 메시지의 다음 홉 노드에 표시되어 있는지를 체크

하고 만약 그렇다면 메쉬 멤버 테이블의 해당 멀티캐스트 그룹 엔트리를 수정하고 그 멀티캐스트 그룹의 메쉬 노드가 된다. 또한, 자신이 다음 홉 노드로 표시되어 있는 코어 송신원들과 이들 코어 송신원으로의 다음 홉 노드 정보를 JAD 메시지에 기록한 뒤, 이 수정된 JAD 메시지를 전송한다. 그림 5는 코어-수신원 메쉬가 형성되는 예를 보여준다.

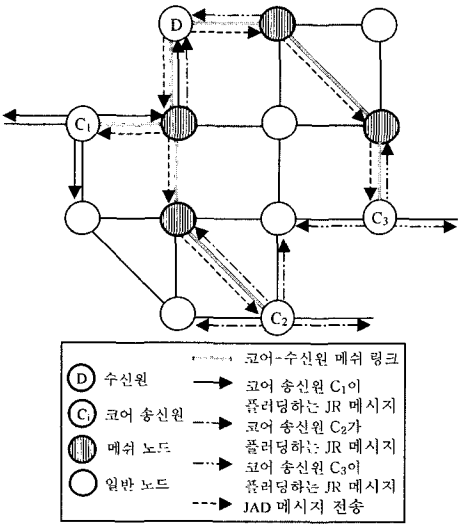

그림 5 코어-수신원 메쉬 형성

이와 같은 과정을 통해 메쉬 노드로 선출된 노드들은 일정 기간(MN-LIFETIME) 동안 해당 멀티캐스트 그룹의 데이타 패킷을 전달하는 역할을 담당한다. 만약 MN-LIFETIME이 지나도록 JAS나 JAD 메시지를 다시 받지 못하면 더 이상 해당 멀티캐스트 그룹의 메쉬 노드로서 동작하지 않게 된다.

코어 송신원은 멀티캐스트 그룹에서 탈퇴하고자 할 때 자신에게 데이타 전달을 의뢰하고 있는 송신원들이 새로운 코어 송신원으로 가는 메쉬를 구성할 때까지 기다려야 한다. 이를 위해, 코어 송신원은 서버에게 LN 메시지를 내보낼 때 코어 테이블의 타이머를 세팅하고, 타이머가 만기될 때까지 데이타 전달 역할을 계속한다. 코어 테이블의 타이머 길이는 서버가 코어 송신원의 LN 메시지를 받고 CoreN 메시지를 플러딩하여 탈퇴하고자 하는 코어 송신원에 연결되어 있던 송신원들이 새로운 송신원-코어 메쉬를 재구성하는 시간을 모두 포함할 수 있도록 충분히 길어야 한다. 멀티캐스트 그룹에서 탈퇴하고자 하는 코어 송신원은 코어 테이블의 타이머가 만기되고 서버로부터 LN 메시지에 대한 응답인 LA 메시지를 받을 때까지 멀티캐스트 그룹에서 떠나지 않고 코어 송신원으로서의 역할을 계속한다. 단, LN 메시지를 내보낸 후에는 추가적으로 자신에게 데이타 전달을 의뢰하는 송신원이 없도록 하기 위해 JR 메시지를 발생시키지 않는다.

2.2 국부적인 메쉬 재구성

앞의 2.1절에서 설명한 주기적인 메쉬 재구성과 더불어 SMMRP는 각 메쉬 노드에서 지속적으로 국부적인 메쉬 재구성을 수행한다. 이 국부적인 메쉬 재구성은 [13,14]에서 제안한 국부적인 메쉬 재구성 방안을 약간 수정한 것이다. 국부적인 메쉬 재구성을 위하여 송·수신원 및 각 메쉬 노드들은 그림 6(a)와 같은 코어 경로 테이블을 유지한다. 코어 경로 테이블에는 해당 노드가 연결되어 있는 각 코어 송신원 주소와 그 코어 송신원으로 연결해주는 상위 메쉬 노드 주소 및 해당 노드로부터 코어 송신원까지의 거리를 기록하고 있다. 이와 같은 정보는 JAS 혹은 JAD를 전달할 때 기록한다. 송·수신원 및 각 메쉬 노드들은 MAC 계층에서 제공하는 비콘 신호를 이용하여 코어 경로 테이블에 기록되어 있는 상위 메쉬 노드들이 전파 전송 범위 안에 있는지를 체크하고, 만약 상위 메쉬 노드가 전파 전송 범위에서 벗어났다면 그 상위 메쉬 노드를 통해 연결되어 있었던 코어 송신원에게 연결해주는 새로운 경로를 찾게 된다. 이러한 국부적인 메쉬 재구성은 요구 단계, 응답 단계, 연결 단계로 진행된다.

코어 송신원으로의 상위 메쉬 노드를 잃어버린 송·수신원 혹은 메쉬 노드는 먼저 가까운 이웃 노드들에게 그림 8의 (1)과 같이 ADVT 메시지(그림 7(a))를 플러딩하는 요구 단계에 들어간다. ADVT 메시지는 잃어버린 상위 메쉬 노드를 통해 연결되어 있었던 코어 송신원 주소, 그 코어 송신원까지의 홉 수 등의 정보를 알린다. 이 때 잃어버린 상위 메쉬 노드를 통해서 하나 이상의 코어 송신원에게 연결되어 있었다면 코어 송신원 주소와 그 코어 송신원까지의 홉 수 정보를 목록으로 작

멀티캐스트 그룹 ID	코어 송신원 주소$_1$	상위 메쉬 노드 주소$_1$	코어 홉 카운트$_1$
	코어 송신원 주소$_2$	상위 메쉬 노드 주소$_2$	코어 홉 카운트$_2$

(a) 코어 경로 테이블

멀티캐스트 그룹 ID	코어 송신원 주소	코어 홉 카운트	PATCH 생성지 주소

(b) PATCH 캐쉬

그림 6 국부적인 메쉬 재구성에 관련된 SMMRP의 자료구조

성한다. ADVT 메시지는 전달될 수 있는 최대 홉 수(그림 7(a)의 TTL 필드)를 2~3으로 제한하여 제한적인 지역 범위에만 플러딩되도록 한다. ADVT 메시지를 받은 노드는 먼저 메쉬 멤버 테이블(그림 2(a))에서 자신이 해당 멀티캐스트 그룹의 메쉬 노드인지 체크하고 그렇다면 다시 코어 경로 테이블(그림 6(a))에서 자신이 ADVT 메시지에 기록되어 있는 코어 송신원으로 가는 경로 상에 있는 메쉬 노드인지를 체크한다. 또한, 자신으로부터 그 코어 송신원까지 홉 수가 ADVT 메시지에 기록된 코어 송신원까지의 홉 수보다 작은지를 체크하는데 이는 ADVT 메시지를 만든 노드보다 코어 송신원에 더 가까운 메쉬 노드들만이 ADVT 메시지에 대한 응답 메시지를 보내도록 하기 위함이다. 만약 ADVT 메시지를 받은 메쉬 노드에서 위의 세 가지 조건이 모두 만족된다면, 그 메쉬 노드는 더 이상 ADVT 메시지를 플러딩하지 않고 ADVT 메시지에 대한 응답 메시지인 PATCH 메시지를 만들어서 그림 8의 (2)와 같이 ADVT 메시지를 만든 노드에게 전송하는 응답 단계에 들어가게 된다. 세 가지 조건 중 하나라도 만족하지 못하는 경우에는 ADVT 메시지의 TTL 필드 값을 하나 감소시키고 해당 메시지를 다시 플러딩한다. 만약 ADVT 메시지의 TTL 필드 값이 0이 되면 더 이상 그 메시지를 전달하지 않는다. ADVT 메시지가 전송되는 동안 ADVT 메시지를 받은 중간 노드들은 ADVT 메시지를 만든 메쉬 노드로 가는 라우팅 정보를 기록하여 이후 ADVT 메시지에 대한 응답인 PATCH 메시지를 라우팅하는데 이를 이용할 수 있도록 한다.

PATCH 메시지(그림 7(b))를 생성하는 메쉬 노드는 PATCH 메시지의 코어 송신원 주소 필드에 ADVT 메시지에서 요구하는 코어 송신원 주소를 적고, 자신으로부터 그 코어 송신원까지의 홉 수를 코어 홉 카운트 필드에 기록한다. 또한, PATCH 메시지의 홉 카운트 필드는 0으로 초기화한다. 이 홉 카운트 필드 값은 PATCH 메시지가 전달되면서 각 중간 노드에서 하나씩 증가시키는데, 이렇게 함으로써, ADVT 메시지를 생성한 노드가 PATCH 메시지를 받았을 때 PATCH 메시지에 기록되어 있는 홉 카운트 필드 값과 코어 홉 카운트 값을 더한 값으로 그 PATCH 메시지를 생성한 메쉬 노드를 통해 코어 송신원과 연결할 경우 자신으로부터 코어 송신원까지의 홉 수가 얼마나 되는지 파악할 수 있다. PATCH 메시지가 ADVT 메시지를 만든 노드에게 전달되는 과정에서, 그 경로 상에 있는 노드들은 PATCH 메시지를 만든 메쉬 노드로 가는 라우팅 정보를 기록하여 PATCH 메시지에 대한 응답인 PATCHA 메시지를 전달하는데 이를 이용할 수 있도록 한다. ADVT 메시지를 발생시킨 노드는 여러 개의 PATCH 메시지를 받을 수 있는데, 이들 중에서 가장 가까운 경로를 제공해 주는 PATCH 메시지 정보를 유지하기 위해 그림 6(b)와 같은 PATCH 캐쉬를 둔다. PATCH 캐쉬는 ADVT 메시지를 발생시킨 노드가 연결하려고 하는 코어 송신원 주소, 해당 코어 송신원까지의 거리, 가장 가까운 경로를 제공해주는 PATCH 메시지를 만든 노드 주소를 저장한다. ADVT 메시지를 만든 노드는 PATCH 메시지들을 일정 기간에 걸쳐 받으면서 이들 중 해당 코어 송신원까지 가장 가까운 경로를 제공하는 PATCH 메시지 정보를 PATCH 캐쉬에 유지한다. 그리고, 그림 8의 (3)과 같이 그 PATCH 메시지를 만든 노드에게 PATCHA 메시지를 만들어 보내고, 그 PATCH 메시지를 전달해 준 이전 홉 노드를 코어 송신원으로 가는 경로 상의 새로운 상위 메쉬 노드로 결정한다.

PATCHA 메시지(그림 7(c))가 전달되면서 국부적인 메쉬 재구성의 마지막 단계인 연결 단계에 들어가게 된다. PATCHA 메시지가 PATCH 메시지를 만든 노드에게 전달되는 경로 상에 있는 중간 노드들은 임시 메쉬 노드로 선출되는데, 이들 임시 메쉬 노드는 주기적 메쉬 재구성에 의해 선출된 일반 메쉬 노드와 같은 일을 수행하지만 [13,14]의 실험을 바탕으로 이들의 MN-

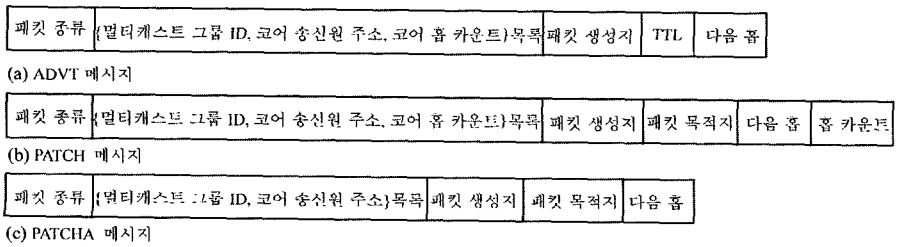

그림 7 국부적인 메쉬 재구성에 관련된 SMMRP의 제어 메시지

LIFETIME 값은 일반 메쉬 노드의 MN-LIFETIME 값의 1/3로 정하였다. 이는 국부적인 메쉬 재구성으로 인해 임시로 선출되는 메쉬 멤버가 지나치게 많아져서 중복 데이타 패킷 오버헤드가 커지는 것을 막기 위함이다.

[13,14]에서 제안한 국부적인 메쉬 재구성 방안에서는 PATCH 메시지에 의하여 임시 메쉬 노드가 선출되기 때문에 ADVT에 응답하여 PATCH 메시지를 보내는 메쉬 노드가 여럿인 경우 임시 메쉬 노드들이 불필요하게 중복적으로 선출된다. 반면에, SMMRP에서는 ADVT 메시지를 만든 노드가 여러 PATCH 메시지들 중에서 코어 송신원으로 가는 가장 짧은 경로를 제공하는 하나의 PATCH 메시지를 선택하고 그 메시지에 대해서만 PATCHA 메시지를 보냄으로써 그 경로 상에서만 임시 메쉬 노드가 선출되도록 한다. 따라서, [13,14]에서 제안된 국부적인 메쉬 재구성 방안과 비교할 때, SMMRP의 국부적인 메쉬 재구성은 더 적은 수의 메쉬 멤버를 선출하고 따라서 중복 데이타 패킷이나 제어 메시지 오버헤드가 더 적다.

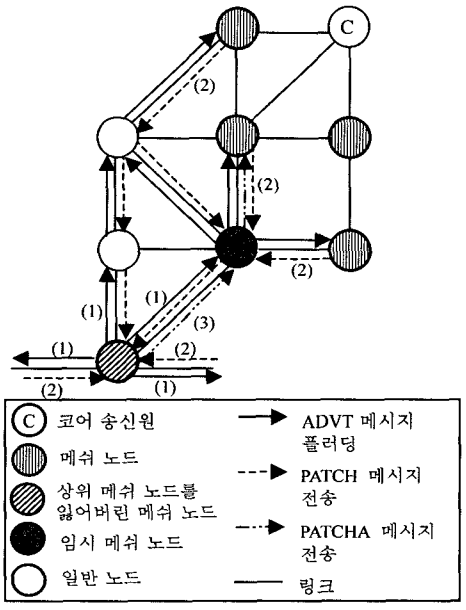

그림 8 국부적인 메쉬 재구성의 예

3. 성능 평가

본 장에서는 시뮬레이션 모델을 설명하고 시뮬레이션 결과를 분석한다. 본 시뮬레이션에서는 제안하는 프로토콜인 SMMRP와 함께, 기존에 제안된 스킴들 중에서 플러딩과 ODMRP의 성능을 비교한다. 시뮬레이션은 Windows NT 4.0 워크스테이션에서 Microsoft Visual C++ 6.0을 사용하였고, UCLA에서 개발한 시뮬레이션 도구인 Global Mobile Simulation(GloMoSim) 라이브러리[15]를 이용하여 구현하였다.

3.1 시뮬레이션 모델

GloMoSim 라이브러리는 각 계층마다 시뮬레이션 모듈을 구현할 수 있는 기능을 제공해준다. 표 1은 본 시뮬레이션에서 사용한 각 계층별 프로토콜을 보여준다.

표 1 시뮬레이션에서 사용한 각 계층별 프로토콜

Layer	Models
Physical (Radio propagation)	Free space with a threshold cut-off
Data link (MAC)	802.11 (CSMA/CA)
Network (Routing)	IP with SMMRP/ODMRP/Flooding
Application	CBR

본 시뮬레이션의 네트워크 모델은 2500 m×2500 m 영역에 무작위로 배치된 180개의 노드들로 구성되며, 이 180개의 노드들 중에서 멀티캐스트 그룹의 송신원과 수신원들은 임의로 선택한다. 선정된 멀티캐스트 그룹 멤버들은 시뮬레이션이 시작할 때 멀티캐스트 그룹에 참여하고, 시뮬레이션이 끝날 때까지 멀티캐스트 그룹의 멤버쉽을 유지한다고 가정한다. 응용에서는 1초당 1개의 패킷을 생성하며, 각 패킷의 크기는 512bytes이다. 각 호스트의 전파 전송 범위는 250m이고 채널 용량은 2Mbps라고 가정한다. 노드의 움직임 모델로는 Random waypoint를 사용한다. 이 모델에 의하면 각 노드는 임의로 목적지를 선택하고 정해진 속도로 그 목적지를 향해 이동한다. 일단, 그 노드가 해당 목적지에 도달하면 0에서 10초 사이의 일정 시간동안 거기에 머물러 있게 된다. 이러한 동작이 시뮬레이션 동안 반복되고, 이때 각 노드의 이동 속도는 15km/h에서 75km/h까지 변화시켜 준다. 각 실험의 시뮬레이션 시간은 1000초이고, 결과 값은 실험의 초기 값을 바꿔주면서 여러 번 실험한 결과의 평균값이다.

3.2 시뮬레이션 결과 및 분석

본 실험에서는 송신원 수에 대한 코어 송신원 수의 비율, 수신원 크기, 주기적인 메쉬 재구성 간격 변화에 따른 SMMRP의 성능을 평가하고자 한다. 성능 평가의 기준치로는 IETF MANET 워킹그룹에서 프로토콜 평가를 위해 제안한 기준[16]에 따라, 평균 데이타 전달율

과 데이타 패킷 오버헤드 및 제어 메시지 오버헤드 등의 세 가지 값을 측정하였다. 구체적으로 이들 세 가지 값은 다음의 식 (1)~식 (3)에 의하여 측정하였는데, 이들 식 중 '각 수신원이 받은 데이타 패킷 수'를 세는데 있어서는 수신원이 한 패킷을 중복으로 여러 개를 받은 경우 한 번만 세었고, '네트워크 전체에 발생한 데이타 패킷(혹은 제어 메시지) 수'는 네트워크 각 노드에서 데이타 패킷(혹은 제어 메시지)를 전달한 횟수를 세었다.

$$\cdot \text{평균 데이타 전달율}(A) = \frac{\sum_{n} R}{\sum_{m} S} \quad (1)$$

$$\cdot \text{데이타 패킷 오버헤드}(B) = \frac{TD}{\sum_{n} R} \quad (2)$$

$$\cdot \text{제어 메시지 오버헤드}(C) = \frac{TC}{\sum_{n} R} \quad (3)$$

m: 송신원 수
n : 수신원 수
S : 각 송신원이 발생시킨 데이타 패킷 수
R : 각 수신원에게 전달된 데이타 패킷 수
TD : 네트워크 전체에 발생한 데이타 패킷 수
TC : 네트워크 전체에 발생한 제어 메시지 수

3.2.1 송신원 수에 대한 코어 송신원 수의 비율 변화에 따른 성능 변화

송신원 수가 증가함에 따라 코어 송신원 수의 변화가 성능에 어떠한 영향을 미치는지를 보기 위하여 송신원 수가 각각 3개, 7개, 12개인 경우에 대하여 표 2와 같이 코어 송신원의 수를 변화시켜 보았다. 이때, 수신원의 수는 10개로 고정시켰으며, 주기적인 메쉬 재구성 간격은 15초로 고정하였다.

표 2 송신원 수와 각각의 경우에 선출된 코어 송신원 수

송신원 수 (개)	3	7	12
코어 송신원 수 (개)	1, 2, 3	1, 2, 4, 7	1, 3, 4, 6, 8, 12

그림 9, 그림 10, 그림 11은 각각 송신원 수가 3, 7, 12인 경우에 대하여 코어 송신원 수를 변화시킬 때의 평균 데이타 전달율을 보여준다. 송신원 수가 비교적 적은 3개인 경우를 살펴보면 송신원 수에 대한 코어 송신원 수의 비율이 30%, 65%, 100%로 증가함에 따라 평균 데이타 전달율도 점차 높아짐을 볼 수 있다. 그러나, 송신원이 7개인 경우만 해도 송신원 대 코어 송신원 비율이 60% 정도가 될 때까지는 코어 송신원 수가 증가

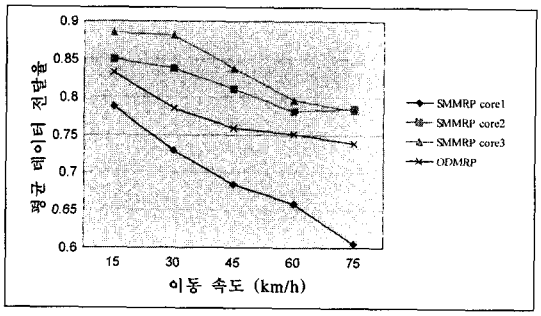

그림 9 송신원이 3개인 경우 코어 송신원 수 변화에 따른 평균 데이타 전달율 변화

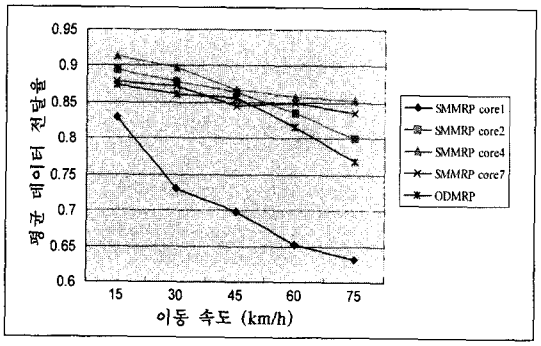

그림 10 송신원이 7개인 경우 코어 송신원 수 변화에 따른 평균 데이타 전달율 변화

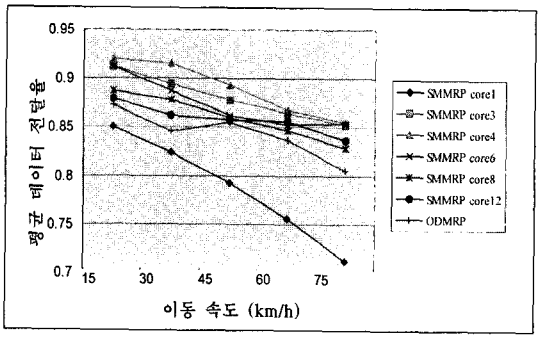

그림 11 송신원이 12개인 경우 코어 송신원 수 변화에 따른 평균 데이타 전달율 변화

함에 따라 평균 데이타 전달율도 높아지지만, 송신원 대 코어 송신원 비율이 100%인 경우에는 60%인 경우에 비하여 오히려 평균 데이타 전달율이 감소하게 됨을 볼 수 있다. 송신원이 12개인 경우에는 이와 같은 현상이 더 일찍 나타나기 시작해 송신원 대 코어 송신원 비율

이 30%보다 더 큰 경우부터 이미 평균 데이타 전달율이 감소하는 것을 볼 수 있다. 이와 같은 결과를 종합하여 볼 때, 송신원 수가 많은 경우 코어 송신원 수가 어느 정도를 넘어서면 오히려 데이타 전달율이 낮아지게 되므로, 송신원 수가 많은 경우에 송신원마다 전달 트리를 형성하는 것이 비효과적임을 확인할 수 있다. 이와 같은 현상은 코어 송신원 수를 늘려주는 것이 성능에 두 가지의 상반되는 영향을 미치는데서 기인한다. 코어 송신원 수가 증가하면 데이타 전달 메쉬가 두텁게 형성되므로, 데이타 전달 구조가 노드의 이동에 더 안정적이게 되는 반면에, 그림 12~17에서 보는 것과 같이 데이타 패킷 오버해드와 제어 메시지 오버헤드가 모두 증가하기 때문에 대역폭 낭비가 커지게 된다. 따라서, 코어 송신원 수가 필요 이상 많아지면 데이타 전달 메쉬가 필요 이상 두터워지고 이로 인한 오버헤드 증가로 안정적인 데이타 전달 메쉬로 인한 이득이 상쇄되어 성능이 감소하게 된다. 또한, 데이타 전달율을 향상시키는 측면에서 유용한 송신원에 대한 코어 송신원 수의 비율이 송신원 수가 증가함에 따라 감소하는 경향이 있음을 볼 수 있는데, 이것은 코어 송신원 수가 어느 정도 이상이 되면 실험한 애드-혹 네트워크 규모에서는 이미 데이타 전달 메쉬가 충분히 조밀하게 형성되어 더 이상 코어 송신원 수를 늘리면 불필요하게 중복 데이타 패킷 및 제어 메시지 오버헤드만 증가되기 때문이다.

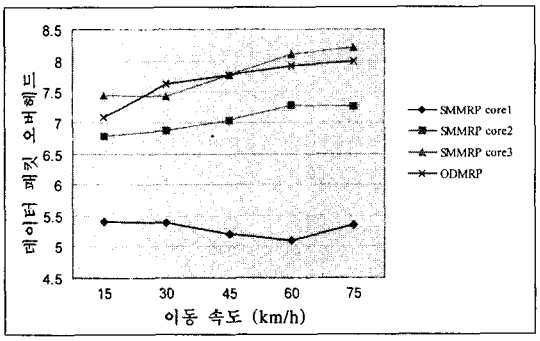

그림 12 송신원이 3개인 경우 코어 송신원 수 변화에 따른 데이타 패킷 오버헤드 변화

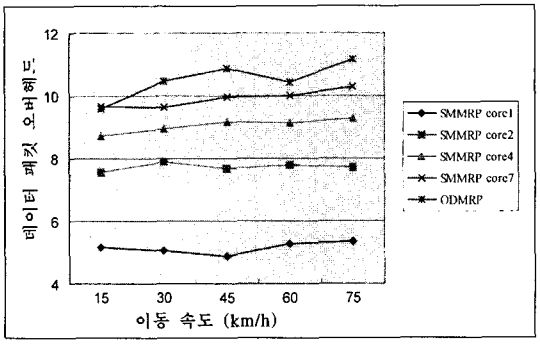

그림 13 송신원이 7개인 경우 코어 송신원 수 변화에 따른 데이타 패킷 오버헤드 변화

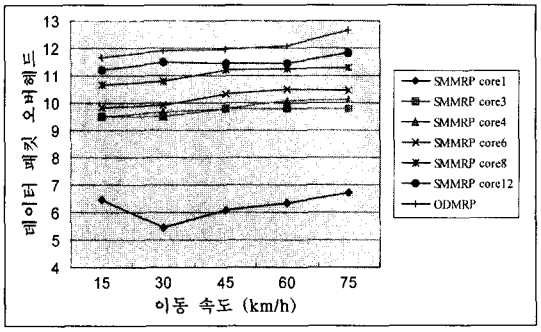

그림 14 송신원이 12개인 경우 코어 송신원 수 변화에 따른 데이타 패킷 오버헤드 변화

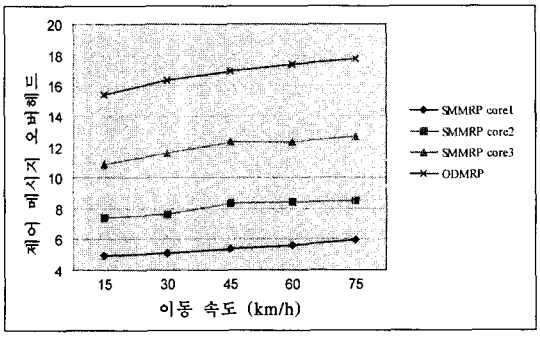

그림 15 송신원이 3개인 경우 코어 송신원 수 변화에 따른 데이타 패킷 오버헤드 변화

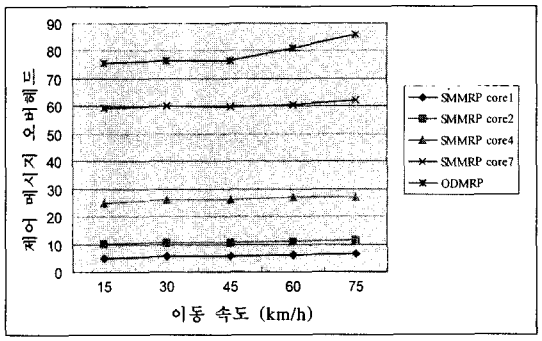

그림 16 송신원이 7개인 경우 코어 송신원 수 변화에 따른 제어 메시지 오버헤드 변화

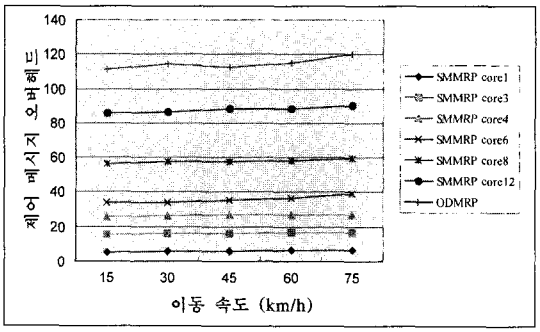

그림 17 송신원이 12개인 경우 코어 송신원 수 변화에 따른 제어 메시지 오버헤드 변화

SMMRP와 ODMRP의 성능을 비교하여 보면, SMMRP가 코어 송신원을 하나만 둔 경우를 제외하고는 모두 SMMRP의 데이타 전달율이 ODMRP보다 나음을 볼 수 있다. 코어 송신원이 한 개인 경우에는 노드의 이동 속도가 커질수록 SMMRP의 평균 데이타 전달율이 급격히 감소하는데, 이것은 코어 송신원이 한 개인 경우에 SMMRP가 구성하는 메쉬 구조가 트리 기반 방식에서 형성되는 데이타 전달 트리와 흡사하기 때문이다. 이 경우, 데이타 전달 메쉬가 노드의 이동성에 잘 대응하지 못하고, 송신원들이 발생시키는 모든 데이타 패킷이 하나의 데이타 전달 경로에 집중되기 때문에 과부하가 발생하게 된다. 노드의 이동성으로 인한 메쉬 단절은 국부적인 메쉬 재구성을 이용하여 메쉬 복구가 이루어지기 때문에 어느 정도 보완이 되지만, 데이타 전달 메쉬에 많은 양의 데이타 패킷이 집중되는 것은 피할 수 없다. 오버헤드 측면에서 보면, 노드 이동 속도에 관계없이 모든 경우에 대해 SMMRP가 ODMRP보다 오버헤드가 더 적으며, SMMRP의 경우 코어 송신원 수가 증가함에 따라 오버헤드도 점차 커지는 것을 볼 수 있다.

SMMRP에서 모든 송신원이 코어 송신원인 경우에는 주기적으로 ODMRP와 유사한 데이타 전달 메쉬를 구성하게 된다. 그런데, 이 경우 데이타 패킷 오버헤드를 보면 SMMRP가 ODMRP보다 약간 낮은 오버헤드를 보이는데, 이것은 지속적인 국부적 메쉬 재구성으로 SMMRP의 데이타 전달율이 더 높기 때문에 (식2)의 데이타 패킷 오버헤드 측정 방법에 의해 이와 같은 결과가 나온 것이다. 즉, 본 논문에서의 데이타 패킷 오버헤드는 제대로 수신된 패킷 하나 당 발생한 중복 패킷이 몇 개인지를 세는 것이기 때문에 SMMRP와 ODMRP에서 각각 발생한 중복적 데이타 패킷 수의 절대량은 이 경우 흡사하였으나, 제대로 전달된 데이타 패킷 수가 SMMRP에서 더 많기 때문에 이와 같은 결과를 보이는 것이다. SMMRP에서 모든 송신원이 코어 송신원인 경우의 제어 메시지 오버헤드를 비교해 보면 데이타 패킷 오버헤드의 경우와 유사하게 제대로 전달된 데이타 패킷 당 제어 메시지 수가 훨씬 적을 뿐 아니라, 제어 메시지의 경우는 절대적인 제어 메시지 양도 SMMRP에서 더 적게 발생한다. 그 이유는 3.2.3의 실험 결과에서 보게 되듯이 SMMRP는 지속적인 국부적 메쉬 재구성을 사용하기 때문에 ODMRP와 같이 짧은 주기적 메쉬 재구성 간격을 사용하는 것이 비효율적이어서 ODMRP보다 3배정도 긴 주기적 메쉬 재구성 간격을 사용하기 때문이다.

평균 데이타 전달율과 오버헤드를 종합적으로 비교하면, SMMRP가 더 적은 오버헤드로 더 높은 데이타 전달율을 제공할 수 있음을 알 수 있다. 또한, SMMRP에서 가장 높은 전달율을 보이는 경우의 코어 송신원 수에 대한 실험 결과를 가지고 SMMRP와 ODMRP를 비교하면 SMMRP가 항상 ODMRP보다 높은 데이타 전달율을 보일 뿐 아니라, 이들 두 스킴의 오버헤드 차이가 송신원 수가 늘어남에 따라 더 커짐을 볼 수 있다. 즉, SMMRP의 경우 송신원 수 증가에 대한 프로토콜의 확장성이 더 좋음을 확인할 수 있다.

3.2.2 수신원 수 변화에 따른 성능 변화

그림 18~20은 수신원 수 변화에 따른 프로토콜 성능의 변화를 비교한 것이다. 이 실험에서는 송신원 수를 12개로 고정하였고, 수신원 수를 5개, 10개, 15개, 20개, 25개, 30개로 변화시켜 보았다. SMMRP의 경우, 4개의 코어 송신원을 두었으며 주기적인 메쉬 재구성 간격을 15초로 설정하였다. 또한, 이 실험에서의 노드 이동 속도는 30 km/h로 설정하였다.

그림 18에서 볼 수 있듯이, 플러딩은 수신원 수에 관계없이 거의 일정한 데이타 전달률을 보이는 반면,

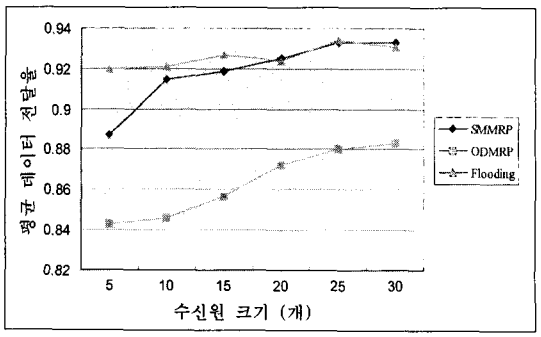

그림 18 수신원 수 변화에 따른 평균 데이타전달률 변화

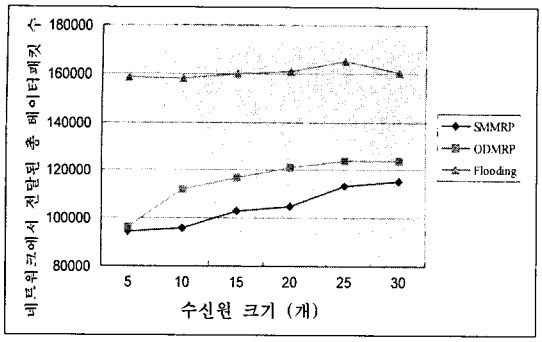

그림 19 수신원 수 변화에 따른 데이타 패킷 오버헤드 변화

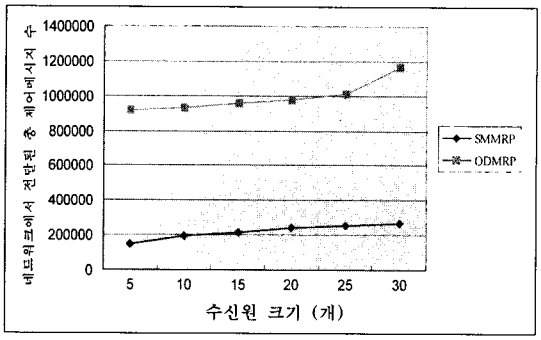

그림 20 수신원 수 변화에 따른 제어 메시지 오버헤드 변화

SMMRP와 ODMRP는 모두 수신원 수가 많아지면 데이타를 포워딩하기 위해 선출되는 메쉬 노드의 수가 증가하게 되어 평균 데이타 전달율이 높아진다. SMMRP의 경우 ODMRP보다 메쉬 노드로 선출되는 노드 수가 적음에도 불구하고 지속적인 국부적 메쉬 재구성으로 인해 ODMRP보다 데이타 전달율이 더 높음을 볼 수 있다.

오버헤드 측정에 있어 이 실험에서는 앞의 식 (2)와 식 (3)에서 설명한 메트릭 대신 발생한 오버헤드 절대량을 보였다. 이 실험에서는 수신원 수 변화에 따라 오버헤드로 인한 네트워크의 부담이 얼마나 증가하는가를 보이고자 하는데, 식 (2)와 식 (3)을 사용하면 수신원 수가 늘어남에 따라 분모가 늘어나서 이와 같은 오버헤드를 비교하기에는 적합하지 않기 때문이다. 그림 19를 보면, 플러딩의 경우에는 SMMRP와 ODMRP에 비하여 훨씬 많은 수의 데이타 패킷이 네트워크에 발생하며, 그 양은 수신원 수에 관계없이 거의 일정함을 알 수 있다. 반면 SMMRP와 ODMRP는 수신원 수가 증가함에 따라 네트워크 상에 발생되는 데이타 패킷 수도 증가하는 것을 볼 수 있는데, 이는 두 스킴 모두 수신원 크기가 커질수록 메쉬 규모가 더 커지게 되기 때문이다. 또한 SMMRP의 네트워크에서 전달된 데이타 패킷 수가 ODMRP의 데이타 패킷 수보다 적은 것을 볼 수 있는데, 이는 SMMRP가 ODMRP보다 더 적은 수의 트리를 통해서 데이타를 내보내기 때문이다.

그림 20에서 네트워크에서 전달된 총 제어 메시지 수를 비교해 보면, SMMRP가 발생시키는 제어 메시지 수가 ODMRP의 1/4 정도이다. 이는 SMMRP에서 코어-수신원 메쉬를 주기적으로 재구성하는 간격이 ODMRP의 경우보다 길기 때문에 SMMRP에서 주기적인 메쉬 재구성을 위해 발생하는 제어 메시지 수가 ODMRP보다 적기 때문이다. 또한, 두 스킴 모두 수신원 수가 많아짐에 따라 제어 메시지 수가 약간 증가하는데, 이것은 수신원 수가 많아질수록, 두 스킴에서 모두 메쉬를 구성하고 유지하기 위해 수신원이 발생시키는 제어 메시지 수가 증가하기 때문이다.

이들 결과를 종합하여 보면, 수신원 수가 증가함에 따른 SMMRP의 성능 변화 양식 자체는 데이타 전달율과 오버헤드 면에서 모두 ODMRP와 매우 유사하다. 즉, 수신원 수에 대한 SMMRP의 확장성 자체는 ODMRP와 유사하다고 볼 수 있다. 그러나, 절대적인 값 자체를 비교해 보면 SMMRP가 데이타 전달율 및 오버헤드 면에서 모두 훨씬 우수함을 알 수 있다.

3.2.3 메쉬 재구성 간격 변화에 따른 성능 변화

그림 21~23은 SMMRP의 주기적인 메쉬 재구성 간격을 5초에서 90초까지 변화시켜보면서 이에 따른 성능의 변화를 비교한 것이다. 이 실험에서는 각각 12개의 송신원과 10개의 수신원을 두었으며, SMMRP에서는 4개의 코어 송신원을 두었다.

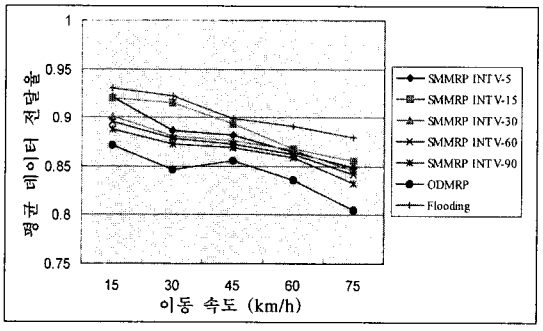

그림 21 주기적인 메쉬 재구성 간격 변화에 따른 평균 데이타 전달율 변화

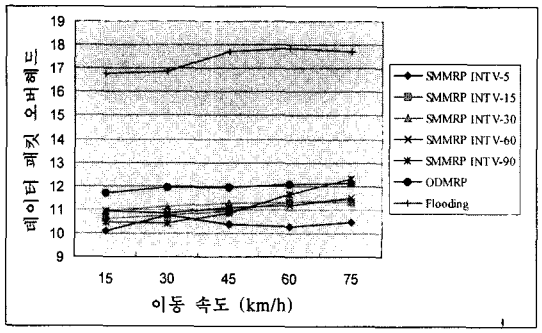

그림 22 주기적인 메쉬 재구성 간격 변화에 따른 데이타 패킷 오버헤드 변화

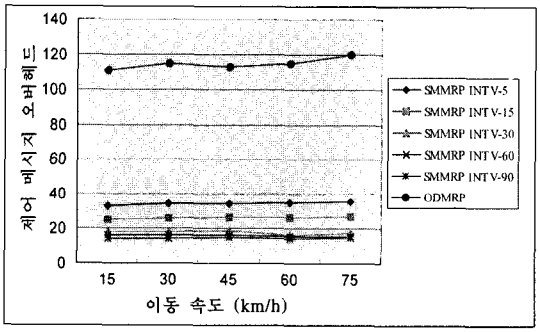

그림 23 주기적인 메쉬 재구성 간격 변화에 따른 제어 메시지 오버헤드 변화

세 가지 스킴들을 비교해볼 때, 3.2.2에서의 결과와 유사하게 플러딩이 가장 높은 데이타 전달율을 보이지만(그림 21), 플러딩은 다른 두 스킴의 2배에 가까운 데이타 패킷 오버헤드를 발생시키기 때문에(그림 22) 애드-혹 네트워크의 트래픽 부하가 큰 경우에는 다른 두 스킴보다 훨씬 큰 성능 감소를 보일 것이다. SMMRP의 경우, 주기적인 메쉬 재구성 간격이 커짐에 따른 데이타 전달율 저하가 그다지 크지 않으며, 주기적 메쉬 재구성 간격이 90초인 경우에도 ODMRP보다 더 높은 데이타 전달율을 보이는데, 이것은 SMMRP가 지속적인 국부적 메쉬 재구성을 사용하기 때문이다. 특히 SMMRP의 주기적 메쉬 재구성 간격이 5초인 경우보다 15초인 경우에 더 데이타 전달율이 높은 것을 볼 수 있는데, 이것은 5초의 경우 제어 메시지 플러딩 오버헤드로 인한 대역폭 낭비가 커서 빈번한 제어 메시지 플러딩을 통한 최적의 메쉬 유지로 얻는 이점을 상쇄하기 때문이다.

데이타 패킷 오버헤드를 보면, 대부분의 경우 SMMRP의 오버헤드가 ODMRP보다 적다. 이것은 SMMRP의 데이타 전달 메쉬가 ODMRP보다 더 적고, SMMRP의 국부적 메쉬 재구성이 효율적으로 이루어지기 때문이다. 실험한 경우 가운데, 이동 속도가 가장 빠르고(75 km/h) 주기적인 메쉬 재구성 간격이 가장 긴(90초) 한 경우에 대해서만 SMMRP의 데이타 패킷 오버헤드가 ODMRP보다 약간 더 큰데, 이것은 이동 속도가 75 km/h와 같이 빠른 경우에는 메쉬 단절이 자주 발생하게 되는데 주기적인 메쉬 재구성 간격이 90초와 같이 길다면 매우 빈번하게 국부적인 메쉬 재구성으로 메쉬 연결이 유지되기 때문이다. 국부적인 메쉬 재구성에 의해서는 최적의 메쉬가 형성되지 못하기 때문에, 이로 인해 데이타 패킷 오버헤드가 커진다. 서로 다른 주기적인 메쉬 재구성 간격에 대한 SMMRP의 성능을 비교해 보면, 데이타 전달율의 측면에서 보았을 때와 마찬가지로 데이타 패킷 오버헤드 측면에서도 주기적 메쉬 재구성 간격이 5초인 경우보다 15초인 경우 더 유리함을 볼 수 있다.

제어 메시지 오버헤드를 보면, 실험한 모든 주기적 메쉬 재구성 간격에 대하여 SMMRP가 발생시키는 제어 메시지 오버헤드가 ODMRP의 1/3 이하이다. SMMRP의 제어 메시지 오버헤드는 주기적인 메쉬 재구성 간격이 커짐에 따라 줄어들지만 그 간격이 늘어나는 비율에 비례해서 감소하지는 않는데, 이것은 SMMRP의 제어 메시지 오버헤드에는 주기적인 메쉬 재구성을 위한 제어 메시지뿐 아니라 국부적인 메쉬 재구성을 위한 제어 메시지도 포함되기 때문이다. 즉, 주기적인 메쉬 재구성을 위한 제어 메시지 오버헤드는 주기적인 메쉬 재구성 간격이 커지는 비율에 따라 비례하여 줄어들지만 국부적인 메쉬 재구성을 위한 제어 메시지는 주기적인 메쉬 재구성 간격이 커질수록 더 많이 발생하는 경향이 있기 때문이다.

이와 같은 결과들을 종합하여 볼 때, SMMRP의 경우 짧은 메쉬 재구성 간격을 유지하여 최적의 데이타 전달 구조를 유지하는 것과 이를 위한 오버헤드 간에 적정한 균형을 유지하는 것이 필요함을 알 수 있다. 또한, SMMRP의 국부적 메쉬 재구성이 주기적 메쉬 재구성을 매우 효과적으로 보완할 수 있음을 알 수 있다.

4. 결 론

본 논문에서는 애드-혹 네트워크에서의 다중점 대 다중점 통신을 효율적으로 지원하기 위해 송신원 수가 많은 경우를 위한 확장성 있는 멀티캐스트 라우팅 프로토콜인 SMMRP를 제안하였다. SMMRP는 멀티캐스트 그룹의 송신원 중 하나 이상의 송신원을 코어 송신원으로 선출하고, 각 코어 송신원을 루트로 하여 멀티캐스트 그룹의 모든 수신원에게 이르는 트리들의 합집합으로

데이타 전달 구조를 형성한다. 코어 송신원이 아닌 송신원들은 각각 자신에게서 가장 가까운 코어 송신원을 선택하여 데이타 전달을 의뢰한다. 데이타 전달 메쉬는 기본적으로 최적의 전달 메쉬를 유지하기 위하여 주기적으로 재구성되지만, 지속적인 국부적 메쉬 재구성에 의하여 연결성이 최대한 유지되도록 한다. SMMRP는 적정한 수의 코어 송신원들을 선출함으로써, 데이타 전달 구조가 필요 이상 비대해지지 않도록 하고 데이타 전달 구조 유지를 위한 오버헤드와 노드 이동성에 대한 안정성간에 균형이 이루어지도록 할 수 있다.

시뮬레이션에 의하여 송신원 수가 많아지는 경우 모든 송신원에 대한 송신원별 트리의 합집합으로 데이타 전달 메쉬를 구성하는 ODMRP보다 제안하는 SMMRP의 성능이 데이타 전달율 및 오버헤드 측면에서 모두 더 나음을 확인할 수 있었다. 또한, SMMRP의 국부적인 메쉬 재구성에 의하여 덜 빈번한 주기적 메쉬 재구성을 효율적으로 보완할 수 있다는 것과, 성능 면에서 최적의 데이타 전달 메쉬를 유지하는 것과 이를 위한 오버헤드 줄이는 것간에 균형을 취해야 함을 알 수 있었다. 향후 연구로는 애드-혹 네트워크 규모 및 멀티캐스트 그룹의 특성과 부하에 따라 서버가 적정 수준의 코어 송신원 비율을 동적으로 결정할 수 있는 방법에 대하여 살펴보고자 한다.

참 고 문 헌

[1] K. Obraczka and G. Tsudik, "Multicast routing issues in ad hoc networks," Proceedings of the IEEE ICUPC98, vol. 1, pp.751~756, Oct. 1998.

[2] C.W. Wu and Y.C. Tay, "AMRIS: A Multicast Protocol for Ad hoc Wireless Networks," Proceedings of MILCOM99, vol. 1, pp.25~29, Nov. 1999.

[3] C.W. Wu, Y.C. Tay and C.K. Toh, "Ad hoc Multicast Routing protocol utilizing Increasing id-numberS(AMRIS) Functional Specification," Internet-Draft, draft-ietf-manet-amris-spec-00.txt, Nov. 1998.

[4] E. Bommaiah, M. Liu, A. McAuley, and R. Talpade, "AMRoute: Ad-hoc Multicast Routing Protocol," Internet-Draft, draft-talpade-manet-amroute-00.txt, Aug. 1998.

[5] E.M. Royer and C.E. Perkins, "Multicast Ad hoc On Demand Distance Vector(MAODV) Routing," Internet-Draft, draft-ietf-manet-maodv-00.txt, Jul. 2000.

[6] E.M. Royer and C.E. Perkins, "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol," Proceedings of Mobicom99, pp. 207~218, Aug. 1999.

[7] S.J. Lee, W. Su and M. Gerla, "On-Demand Multicast Routing Protocol(ODMRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-odmrp-02.txt, Jul. 2000.

[8] S.J. Lee, M. Gerla and C.C. Chiang, "On-Demand Multicast Routing Protocol," Proceedings of IEEE WCNC99, vol. 3, pp.1298~1302, Sep. 1999.

[9] J.J. Garcia-Luna-Aceves and E.L. Madruga, "Scalable Multicasting: The Core-Assisted Mesh Protocol," ACM/Baltzer Mobile Networks and Applications Journal, Special Issue on Management of Mobility in Distributed Systems, vol. 6, pp.151~165, Apr. 2001.

[10] J.J. Garcia-Luna-Aceves and E.L. Madruga, "The Core-Assisted Mesh Protocol," IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp.1380~1394, Aug. 1999.

[11] S.J. Lee and C.W. Kim, "Neighbor Supporting Ad hoc Multicast Routing Protocol," Proceedings of MobiHOC2000, pp.37~44, Aug. 2000.

[12] T. Ozaki, J.B. Kim and T. Suda, "Bandwidth-Efficient Multicast Routing for Multihop, Ad-Hoc Wireless Networks," Proceedings of IEEE INFOCOM2001, vol. 2, pp.1182~1191, Apr. 2001.

[13] 김예경, 이미정, "이동성이 큰 애드-혹 네트워크를 위한 효율적인 멀티캐스트 라우팅 프로토콜", 한국정보과학회 논문지, 정보통신 제28권, 제1호, pp.143~153, Mar. 2001.

[14] M.J. Lee and Y.K. Kim, "PatchODMRP: An Ad-hoc Network Multicast Routing Protocol," Proceedings of ICOIN-15, 제 15 권, pp.537~543, Feb. 2001.

[15] Wireless Adaptive Mobility Lab. Dept. of Comp. Sci., UCLA "GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems," http://pcl.cs.ucla.edu/projects/glomosim

[16] M.S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," Request For Comments 2501, Internet Engineering Task Force, Jan. 1999.

강 현 정
1996년~2000년 이화여자대학교 컴퓨터학 학사. 2000년~2002년 이화여자대학교 컴퓨터학 석사. 2002년~현재 삼성전자 통신연구소 표준연구팀 연구원. 주관심분야는 Ad hoc network, 모바일 컴퓨팅

이 미 정
정보과학회논문지 : 정보통신
제 30 권 제 2 호 참조