애드 혹 네트워크에서 클러스터를 이용한 효율적인 플러딩 방안

(An Efficient Flooding Scheme using Clusters in Mobile Ad-Hoc Networks)

왕 기 철 † 김 태 연 †† 조 기 환 †††
(Gi-cheol Wang) (Tae-yeon Kim) (Gi-hwan Cho)

요 약 애드 혹 네트워크에서 플러딩(flooding)은 통신범위내에 있지 않은 노드로의 다중 홉 경로를 찾기 위해 자주 이용된다. 그러나 기존의 플러딩기법들은 주기적인 메시지 교환, 잦은 충돌(collision)발생, 그리고 불필요한 패킷의 방송으로 인해 네트워크의 성능을 저하시킨다. 이러한 문제점을 해결하기 위해 보다 새롭고 효율적인 플러딩방법이 요구된다. 본 논문에서는 요구형 클러스터 구성을 이용하여 플러딩을 수행하는 기법을 제안한다. 제안하는 방법은 기존의 요구형 클러스터 구성기법과 같이 지나가는 패킷(ongoing packet)을 이용하여 클러스터를 구성한다. 그러나 제안하는 방법은 기존의 요구형 클러스터 구성기법과는 다르게 유니캐스트 패킷 전송을 이용하여 충돌횟수를 줄이고 이웃 플러딩 후보들을 용이하게 파악한다. 따라서 제안하는 방법은 다른 기법들에 비해 보다 작은 수의 플러딩노드를 생성한다. 실험결과들은 제안하는 방법이 다른 기법들에 비해 패킷전송 횟수 및 충돌횟수를 감소시킴을 입증하였다.

키워드 : 애드 혹 네트워크, 플러딩, 클러스터 생성

Abstract Flooding is usually utilized to find a multi hop route toward the destination which is not within transmission range in Ad Hoc networks. However, existing flooding schemes deteriorate the network performance because of periodic message exchanges, frequent occurrence of collisions, and redundant packet transmission. To resolve this, a flooding scheme using on demand cluster formation is proposed in this paper. The scheme employs ongoing packets for constructing a cluster architecture as the existing on demand clustering scheme. Unlike to the existing on demand clustering scheme, the scheme makes use of unicast packet transmission to reduce the number of collisions and to find the flooding candidates easily. As a result, the proposed scheme yields fewer flooding nodes than other schemes. Simulation results proved that the proposed scheme reduces the number of transmissions and collisions than those of two other schemes.

Key words : Ad Hoc networks, flooding, cluster formation

1. 서 론

애드 혹 네트워크는 AP(Access Point)나 기지국과 같은 기반구조 없이 이동 단말들 로만 구성된 자율적이고 독립적인 네트워크이다. 애드 혹 네트워크는 구성이 단순하고 융통성 있으며, 일시적인 필요에 의한 임시 네트워크의 구성에 용이하기 때문에, 전쟁터에서의 통신, 재난구조 상황에서의 통신, 그리고 교실이나 회의실에서 즉석통신과 같은 다양한 분야로의 응용이 논의되고 있다.

임의의 애드 혹 네트워크에서 만일 두 노드가 서로의 전송범위 안에 있다면, 그들은 직접 통신을 수행할 수 있다. 그렇지 않은 경우에는 그들은 다른 노드들의 도움을 받아서 통신을 수행해야 한다. 이러한 도우미 노드(helper node)들을 찾기 위해서 대부분의 요구형(on-demand) 라우팅 프로토콜들 [1-3]은 플러딩 메시지들을 이용한다.

플러딩 상황에서 임의의 노드는 임의의 메시지를 자신의 모든 이웃들에게 방송한다. 그 이웃노드들은 다시 그들의 이웃노드들에게 그 메시지를 방송한다. 이러한 방송메시지의 전파는 모든 노드들이 그 메시지를 수신

† 정 회 원 : 전북대학교 영상정보신기술연구소 연구원
gcwang@dcs.chonbuk.ac.kr
†† 정 회 원 : 서남대학교 컴퓨터정보통신학과 교수
tykim@seonam.ac.kr
††† 종신회원 : 전북대학교 전자정보공학부 교수
ghcho@dcs.chonbuk.ac.kr
논문접수 : 2005년 4월 25일
심사완료 : 2005년 9월 29일

할 때까지 반복된다. 이러한 플러딩 방법을 맹목적 플러딩(blind flooding)라 한다. 맹목적 플러딩은 불필요한 패킷의 전송 및 많은 충돌을 유발한다. 특히, 노드간의 연결성이 높거나 노드들이 밀집해 있는 경우에, 이러한 문제점들은 네트워크의 성능을 크게 저하시킨다.

이러한 맹목적 플러딩의 문제점들을 해결하기 위해, 위상정보를 이용하여 플러딩에 참여하는 노드를 줄이는 플러딩기법들 [4-8]이 제안되어 왔다. 이러한 기법들은 위상정보를 획득하기 위해, 주기적인 이웃정보의 교환을 요구한다. 그러나 이동성, 제한된 대역폭 및 자원, 잦은 충돌 발생과 같은 애드 혹 네트워크의 특성으로 인해 임의의 노드가 정확한 위상 정보를 획득하기는 어렵다. 더구나 이러한 기법들은 과도한 통신 오버헤드를 유발한다. 이러한 이유로 대부분의 요구형 라우팅 프로토콜들은 단순히 맹목적 플러딩을 이용해 왔다. 따라서 애드 혹 네트워크에서 주기적인 메시지 교환 없이 플러딩을 수행하는 노드의 수를 감소시키는 플러딩 기법이 요구된다. 이후부터 플러딩에 참여하는 노드를 플러딩 노드로, 그렇지 않은 노드를 비 플러딩노드라 칭하기로 한다.

본 논문은 주기적인 이웃정보의 교환없이 지나가는 패킷(ongoing packet)을 이용하여 클러스터 구조를 생성하고 이 클러스터 구조를 이용하여 효율적으로 플러딩을 수행하는 방법을 제안한다. 주기적인 이웃정보의 교환없는 클러스터 구성방법은 이미 참고문헌 [9]에서 제안되었지만, 임의의 노드는 비 플러딩노드가 되기 위해서 다음과 같은 선행조건들을 만족하여야 한다. 먼저, 임의의 노드는 클러스터 헤드가 아닌 멤버노드여야 한다. 둘째, 그 멤버노드의 이웃에 세 개 이상의 플러딩노드가 존재하여야 한다. 만일 그 멤버노드가 더 많은 이웃 플러딩노드를 가질수록, 그 멤버노드가 비 플러딩노드가 될 확률은 더 높아진다. 이것은 비 플러딩노드들이 이웃의 플러딩노드들 사이에서 선정되기 때문이다. 그럼에도 불구하고, 방송패킷의 잦은 충돌유발로 인해 임의의 노드가 이웃의 플러딩노드들을 충분히 파악하기는 힘들다.

제안하는 방법은 비 플러딩노드가 되기 위한 조건을 완화시켜 임의의 노드가 비 플러딩노드가 될 확률을 높인다. 즉, 제안방법은 멤버노드는 물론 클러스터 헤드도 비 플러딩노드가 되도록 허용한다. 또한 제안방법은 유니캐스트 패킷전송을 통해 임의의 노드가 많은 수의 이웃 플러딩노드를 파악하도록 도와준다. 이러한 이웃 플러딩노드를 이후에는 플러딩 후보라 칭하기로 한다. 이러한 플러딩 후보들 중에서 비 플러딩노드들은 클러스터 정보의 원천지, 클러스터에서의 역할, 그리고 식별자(identifier)의 비교를 통하여 선정된다.

본 논문의 구성은 다음과 같다. 2장에서는 애드 혹 네트워크에서 효율적인 플러딩을 위한 기법들이 소개된다. 3장에서는 제안하는 방법이 자세히 기술된다. 4장에서는 실험결과가 소개되고, 5장은 결론을 내린다.

2. 관련 연구

최근까지 애드 혹 네트워크에서 맹목적 플러딩의 문제점들을 해결하기 위한 다양한 방법들이 제안되어 왔다. 그러나 애드 혹 네트워크에서 플러딩노드들의 최적 집합을 찾는 문제는 NP-complete문제로 알려져 있다[4]. 따라서 기존 연구들 [3-8]은 플러딩노드들의 부분 최적 집합을 찾기 위한 다양한 선험적 방법들을 제안하였다.

참고문헌 [7]에서는 수신자 가지치기 기법(self pruning)과 송신자 가지치기 기법(dominant-pruning)이라는 두 가지 플러딩노드 감소기법을 제안하였다. 수신자 가지치기 기법에서, 각 전송노드는 나가는 패킷(outgoing packet)에 자신의 이웃노드 리스트를 삽입한다. 이 패킷을 수신하는 임의의 노드는 패킷 송신자의 이웃에 포함되지 않은 노드가 이웃을 가진 경우에만 수신패킷을 재전송한다. 반면에, 송신자 가지치기 기법에서는 패킷의 송신자가 패킷을 전송하기 전에 자신의 2홉 이내에 존재하는 모든 노드들을 cover할 수 있는 전송자 리스트(forward list)를 미리 설정한다. 송신자는 이 전송자 리스트를 나가는 패킷에 삽입한다. 패킷과 함께 수신된 전송자 리스트에 포함된 노드만이 수신된 패킷을 재전송한다.

참고문헌 [5]에서의 MPR(Multipoint Relay)방법은 송신자 가지치기 기법과 유사한 전략을 사용하며 다만 전송자 리스트(forward list)를 선정하는 방법만 다르다.

참고문헌 [4,8,9]에서는 클러스터 구성(cluster formation)을 이용하여 플러딩노드의 수를 감소시키는 방법들을 제안하였다. 이동 호스트들을 일련의 기준에 따라 그룹화 하는 것을 클러스터 구성(cluster formation)이라 한다. 임의의 클러스터는 하나의 클러스터 헤드, 멤버노드들로 구성된다. 멤버노드들 중에서 일부노드들은 게이트웨이들이 된다. 여기서 클러스터 헤드는 클러스터의 대표자 역할을 수행하고 멤버노드들은 클러스터 헤드의 서비스를 받는다. 또한 게이트웨이들은 인접 클러스터들을 연결하는 역할을 수행한다. 임의의 클러스터 내에서 클러스터 헤드는 모든 멤버들과 직접 연결되어 있다. 따라서 클러스터 헤드가 데이타를 방송하면 게이트웨이와 멤버노드들은 그 데이타를 동시에 수신할 수 있다. 따라서 클러스터 구조를 통하여 플러딩이 수행된다면, 클러스터 헤드와 하나의 게이트웨이 노드만 플러딩 노드가 되고 나머지 노드들은 비 플러딩노드가 된다. 즉 클러스터내에서 멤버들끼리는 메시지를 전파하지 않게 되므로 임의의 메시지를 네트워크에 전파하는데 필요한 재전송 횟수가 줄어들 것이다.

그러나 위에서 언급한 능동적 클러스터 구성 [4,8]은

추가적인 메시지 교환을 요구한다. 예를 들어 식별자가 클러스터 헤드 선출의 기준이라면, 헬로 메시지교환을 주기적으로 수행해야 한다[10]. 만일 연결성(connectivity)이 클러스터 헤드 선출의 기준이라면, 두 번의 헬로 메시지 교환(식별자 와 연결성)을 주기적으로 수행해야 한다 [10]. 더구나 게이트웨이들 중에서 전송 게이트웨이를 선정하기 위해서 추가적인 메시지 교환이 요구된다. 이러한 추가적인 메시지 교환은 오히려 네트워크의 성능을 크게 저하시킨다.

참고문헌 [9]에서는 주기적인 메시지 교환 없이 플러딩에 참여하는 노드의 수를 감소시키는 요구형 클러스터 구성 방법이 제안되었다. 이 방법은 지나가는 패킷(ongoing packet)을 이용하여 클러스터를 구성하고 비 플러딩노드를 선정한다. 그러나 이 방법에서 임의의 노드는 비 플러딩노드가 되기 위해서 충분한 이웃의 플러딩 노드를 알고있어야 한다. 그러나 방송메시지의 잦은 충돌로 인해 임의의 노드가 충분한 수의 이웃 플러딩 노드를 파악하기는 어렵다. 더구나 이 방법은 임의의 노드가 비 플러딩노드가 되기 위한 조건이 까다롭다. 따라서 이 방법은 비 플러딩노드수의 감소가 일정한도 내에서 제한될 수밖에 없다.

따라서 본 논문에서는 지나가는 패킷을 이용하여 클러스터를 구성하면서도, 참고문헌 [9]에서의 방법 보다 많은 비 플러딩 노드를 생성하는 클러스터 구성 기법을 제안한다.

3. 플러딩노드의 수를 감소시키기 위한 유니캐스트 전송기반의 클러스터 구성

3.1 가정사항 및 준비사항

제안하는 방법을 기술하기에 앞서 다음과 같은 사항들을 가정한다. 첫째, 네트워크내의 모든 노드는 혼합(promiscuous)모드 [1]로 동작한다. 둘째, 각 노드의 MAC(Media Access Control) 프로토콜은IEEE 802.11 DCF(Distributed Coordination Function)이다. 셋째, 만일 임의의 두 노드가 서로의 전송범위 내에 존재한다면, 두 노드 사이에는 양방향 링크가 설정된다.

플러딩에 참여하는 각 노드는 패킷을 전송하기에 앞서 클러스터 상태 정보를 그 패킷에 삽입한다. 지나가는 패킷에 클러스터 정보를 삽입하는 목적은 두 가지 이다. 하나는 지나가는 패킷을 이용하여 클러스터 구조를 구축하는 것이고, 다른 하나는 이 패킷을 이용하여 비 플러딩노드들을 선택하는 것이다. 이러한 목적을 위해 클러스터 상태정보는 다음과 같은 필드로 구성된다.

- 송신자 식별자: 이 클러스터 정보를 전송하는 노드의 IP 주소
- 선행자 식별자: 이 클러스터 정보의 목적지 IP 주소
- 클러스터내의 역할: 이 클러스터 정보를 전송하는 노드의 클러스터내 역할(클러스터 헤드 혹은 게이트웨이)

제안하는 방법에서 소스노드를 제외한 노드로부터 전송되는 패킷은 참고문헌 [8]과는 다르게 모두 유니캐스트 패킷이다. 일반적으로 유니캐스트 패킷은 특정노드를 향하여 전송되지만, 무선의 방송특성으로 인해 이웃에 있는 모든 노드들은 이 패킷을 수신할 수 있다. 이를 위해서는 각 노드의 무선 네트워크 인터페이스는 혼합 모드 동작을 허용하여야 한다. 혼합 모드에서 각 노드의 네트워크 인터페이스 하드웨어는 수신된 모든 패킷을 자신의 네트워크 드라이버 소프트웨어로 전송한다[1]. 수신된 패킷에서의 클러스터 정보를 이용하여 각 노드는 자신의 클러스터 상태를 결정하고 클러스터 정보를 갱신한다. 먼저, 각 노드는 자신의 선행자 식별자를 수신된 클러스터 정보의 송신자 식별자로 설정한다. 또한 각 노드는 자신의 클러스터 상태를 수신된 클러스터 정보의 클러스터 상태에 따라 다음의 네 가지 상태중 하나로 설정한다.

- "초기상태": 어떠한 클러스터에도 가입이 되지 않은 노드는 자신의 상태를 "초기상태"로 설정한다. 초기에 네트워크에 가입하는 모든 노드는 이 상태를 가진다.
- "클러스터 헤드": "비 플러딩노드" 상태가 아닌 노드가 "게이트웨이" 상태를 가진 노드로부터 패킷을 수신하면 노드는 이 상태로 전환한다.
- "게이트웨이": "비 플러딩노드" 상태가 아닌 노드가 "클러스터 헤드" 상태를 가진 노드로부터 패킷을 수신하면 노드는 이 상태로 전환한다.
- "비 플러딩노드": "게이트웨이"나 "클러스터 헤드"상태인 노드가 같은 선행자 식별자 및 클러스터 상태를 가지고 있으면서 더 작은 식별자를 가진 노드를 발견하면, 자신의 상태를 이 상태로 전환한다.

3.2 클러스터 구성 방법

이 절에서는 각 노드가 수신된 패킷을 기반으로 자신의 클러스터 상태를 결정하는 절차를 단계별로 기술한다.

1 임의의 플러딩소스는 클러스터 정보의 송신자 식별자와 선행자 식별자를 자신의 식별자로 그리고 자신의 클러스터 상태를 "클러스터 헤드"로 설정한다. 이후 그 소스는 그 클러스터 정보를 새로운 플러딩패킷에 삽입하고 방송한다.

2 이 패킷을 수신하는 임의의 노드는 그 패킷의 신규성을 점검한다. 신규성은 플러딩소스에 의해 채워지는 소스식별자와 순서번호에 의해 결정된다. 만일 수신 패킷이 새로운 것이면, 그 패킷은 노드의 지역저장소에 저장된다.

2.1 만일 그 노드가 새로운 패킷을 수신했다면, 그 노드는 자신의 선행자와 클러스터 상태를 결정한다. 즉, 선행자 식별자는 수신 클러스터 정보의 송신자 식별자로 설정되고 클러스터내의 역할은 3.1절에서 기술된 것 처럼, 자신의 클러스터 상태와 수

신 패킷의 클러스터내의 역할에 따라 변경된다.

2.2 만일 그 노드가 이미 수신한 패킷을 이웃노드들의 패킷전송으로 인해 다시 수신했다면, 그 노드는 수신패킷의 클러스터 정보를 지역저장소에 저장된 패킷의 클러스터 정보와 비교한다. 만일 선행자 식별자와 클러스터내의 역할필드가 지역저장소에 저장된 패킷과 일치한다면, 그 노드는 자신의 식별자와 방금 수신된 클러스터 정보의 송신자 식별자를 비교한다. 만일 자신의 식별자가 방금 수신된 클러스터 정보의 송신자 식별자보다 크다면, 작은 식별자를 가진 노드를 플러딩노드로 등록하고 자신의 상태를 "비 플러딩노드"로 변경한다. 그렇지 않다면, 방금 수신된 클러스터 정보는 무시된다. 이러한 경합상황에서 작은 식별자를 가지는 노드를 이후에는 후계자라 부르기로 한다.

"비 플러딩노드"상태를 가지는 노드는 다음의 상황하에서 자신의 상태를 "초기상태"로 변경한다. 첫 번째는 임의의 "비 플러딩노드"노드가 정해진 시간내에 자신의 선행자나 후계자로부터 패킷을 수신하지 못하는 경우이다. 두 번째는 임의의 "비 플러딩노드"가 선행자나 후계자가 아닌 노드로부터 패킷을 수신하는 경우이다.

3 단계 2부터 3까지의 반복을 통해 클러스터 구조의 구축 및 비플러딩 노드의 선택을 완료한다.

만일 임의의 "비 플러딩노드"들이 장기간 그 상태를 유지한다면, 노드들의 패킷수신률은 심각하게 손상된다. 그러한 이유로 임의의 "비 플러딩노드"는 주기적으로 자신의 "비 플러딩노드" 자격을 검사하여 자격상실이 되었을 때는 플러딩에 다시 참여하도록 한다. 임의의 노드가 특정기간내에 자신의 선행자나 후계자로부터 패킷을 수신하지 못할 때와 자신의 선행자나 후계자로부터 패킷을 수신하지 못할 때, 그 노드는 자격상실을 감지하게 된다. 이런 이유로 단계 2에서 비 플러딩노드의 패킷 수신체크는 위의 클러스터 구성과정에서 필요하게 된다.

3.3 클러스터 구성 예제

그림 1과 2는 임의의 애드 혹 네트워크에서 제안하는 클러스터 구성 방법에 의해 패킷이 전달되는 과정을 보여준다. 그림 3은 그림 1과 2의 과정을 반복함으로써 완성된 클러스터 구조를 보여준다. 그림 1, 2, 3에서 노드위에 표시된 숫자들은 노드들의 선행자를 나타내고 사각형안의 숫자들은 노드들의 후계자를 나타낸다.

먼저, 그림 1은 플러딩소스의 방송과 선행자로의 유니캐스트 전송을 보여준다. 그림 1에서 플러딩 소스 2는 자신이 클러스터 헤드임을 알리는 패킷(즉, "클러스터 헤드"상태를 가지는 패킷)을 방송하고 이를 수신하는 노드 7, 16, 19는 "게이트웨이"로 상태를 변경하고 노드 2를 그들의 선행자로 설정한다. 또한 그 노드들은 그들의 선행

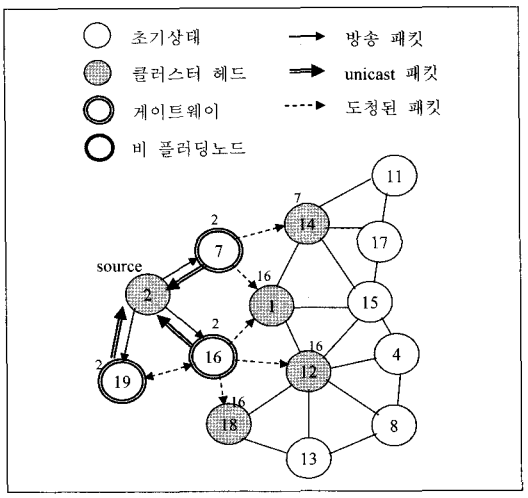


그림 1 플러딩 소스의 방송과 수신패킷 핸들링의 예제

자(즉, 노드 2)에게 전송되는 유니캐스트 패킷에 그들의 클러스터 정보를 첨부한다. 이때 노드 1, 12, 14 그리고 18은 이 패킷을 혼잡모드 수신을 통해 획득하고 획득된 클러스터 정보에 따라 자신들의 상태를 "클러스터 헤드"로 변경한다. 또한 획득된 클러스터 정보의 전송자(즉, 송신자 식별자)를 그들의 선행자로 설정한다. 여기서 노드 1은 노드 7과 16으로부터 두개의 혼잡 패킷을 수신함에 유의해야 한다. 이 경우에 노드들은 먼저 수신된 패킷을 취하고 나중에 수신된 패킷은 버린다. 즉, 노드 1이 16으로부터의 패킷을 먼저 수신한다면, 7로부터 수신되는 패킷은 바로 버린다. 같은 방법으로 노드 12, 14, 18은 노드 16, 7, 16을 각각 그들의 선행자로 설정한다.

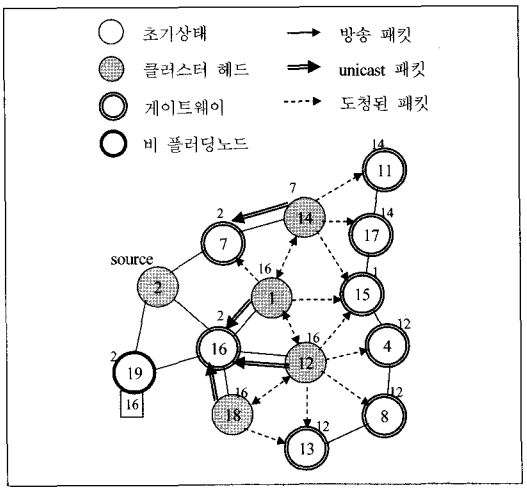


그림 2 수신 패킷 핸들링 및 선행자로의 패킷전송에 관한 예제

그림 2에서, 노드 16과 19는 상대방의 유니캐스트 패킷전송을 통해 상대방의 존재를 인지하고 자신의 상태 변화를 꾀한다. 예를들어 노드 16은 노드 19가 자신과 같은 선행자 및 클러스터내의 역할을 가지고 있다는 것을 알게된다. 그러나 자신의 식별자가 노드 19보다 작으므로 크 클러스터 정보를 무시한다. 반면에, 노드 19는 16보다 큰 식별자를 가지므로, 자신의 상태를 "비 플러딩노드" 로 변경하고 16을 후계자로 설정한다. 이런 와중에 노드 1, 12, 14, 18은 각각 자신의 선행자에게 자신들의 클러스터 정보를 포함하는 유니캐스트 패킷을 전송한다. 노드 4, 8, 11, 13, 15, 그리고 17은 이 패킷 전송을 통해 그들의 클러스터 상태를 "게이트웨이"로 변경하고 노드 12, 12, 14, 12, 1, 그리고 14를 각각 자신들의 선행자로 설정한다.

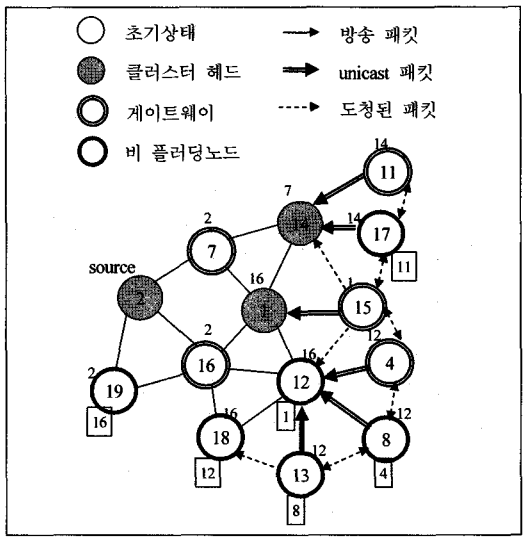


그림 3 제안된 방법을 통한 클러스터 구성 및 비플러딩 노드 선정 완료

노드 4, 8, 11, 13, 15, 그리고 17은 자신들의 클러스터 정보를 포함하는 유니캐스트 패킷을 자신의 선행자에게 전송한다. 이 전송을 통해 이 노드들은 자신의 이웃에 있으면서 같은 선행자 식별자 및 클러스터내의 역할을 가지는 노드들을 파악한다. 그림 3에서 노드 17은 노드 11이 자신과 같은 선행자 식별자 및 클러스터내의 역할을 가진다는 것을 알게 되므로, 자신의 상태를 "비 플러딩노드"로 변경한다. 노드 8과 13도 각각 노드 4와 8이 같은 선행자 식별자 및 클러스터내의 역할을 가진다는 것을 인식하게 되므로, 그들의 상태를 "비 플러딩 노드"로 변경한다. 노드 15는 이웃에 같은 선행자 식별자 및 클러스터내의 역할을 가지는 노드가 없으므로 자신의 "게이트웨이" 상태를 그대로 유지한다.

4. 실험결과

4.1 실험 파라미터 및 메트릭

본 논문에서 제안된 기법의 성능을 분석하기 위하여 시뮬레이터를 이용하여 실험을 수행하였다. 실험도구로는 CMU(Carnegie Mellon University)의 무선 확장 [11]을 구현한 ns-2 네트워크 시뮬레이터 [12]를 이용하였다. 이동 시나리오는 노드수, 정지시간, 최대이동속도, 실험영역등과 같은 입력 파라미터들을 이용하여 랜덤하게 생성하였다. 실험을 위한 파라미터들은 표 1과 같이 설정되었다.

제안하는 방법의 플러딩 효율성을 평가하기 위해 소스들이 플러딩 패킷들을 일정 시간간격으로 전체 네트워크에 전송하는 새로운 응용을 이용한다. 이러한 응용을 가지고 제안하는 방법은 맹목적 플러딩방법과 수동적 클러스터링 방법 [9]과 비교되었다. 각 실험의 시작에서 임의의 단일 플러딩소스가 선정되었다. 또한 각 실험은 600초(10분) 동안 수행되었다.

표 1 시뮬레이션 파라미터들

파라미터	값
노드의 수	100
실험 영역	600㎡ 2000㎡
노드의 최대이동속도	0 16m/s
이동사이의 정지시간	100초
전송범위	250m
패킷방송 간격	0.25초
방송패킷 크기	100byte
이웃인식 타이머 종료간격	0.5초
MAC프로토콜	IEEE 802.11 DCF

제안하는 방법을 가진 플러딩기법의 효율성을 분석하기 위해 다음과 같은 메트릭들을 평가한다.

- 한번의 플러딩 동안 전송되는 평균 패킷의 수: 모든 노드들로부터 전송되는 플러딩패킷의 총수가 소스로부터 발생되는 플러딩패킷의 총수에 의해 나누어 진다.
- 한번의 플러딩동안 수신하는 노드의 평균 비율: 먼저, 소스로부터 발생되는 임의의 플러딩패킷에 대해 이를 수신하는 노드의 수가 측정된다. 이 값이 1이면 모든 노드가 소스로부터의 플러딩패킷을 수신한 것이다. 이 비율들은 시뮬레이션이 끝날 때까지 누적되고 평균비율은 누적된 값을 소스로부터 발생된 방송패킷의 총수에 의해 나누는 것에 의해 구해진다.
- 한번의 플러딩에 대해 발생하는 충돌 횟수: 실험기간 동안 모든 노드로부터 발생하는 충돌의 수가 소스로부터 발생되는 플러딩패킷의 총수에 의해 나누어 진다. 먼저 실험영역을 1000㎡으로 고정시킨 후에 최대 이

동속도를 변화시켜 가면서 위의 메트릭들을 측정하였다. 그후에 최대 이동속도를 0으로 고정시킨 후에 실험영역을 증가시켜 가면서 메트릭들을 측정하였다.

4.2 실험결과

그림 4, 5, 6은 실험영역을 고정시킨 상황에서 노드의 이동속도가 증가함에 따라 전송패킷의 수, 패킷 수신율, 충돌횟수의 변화를 각각 보여준다.

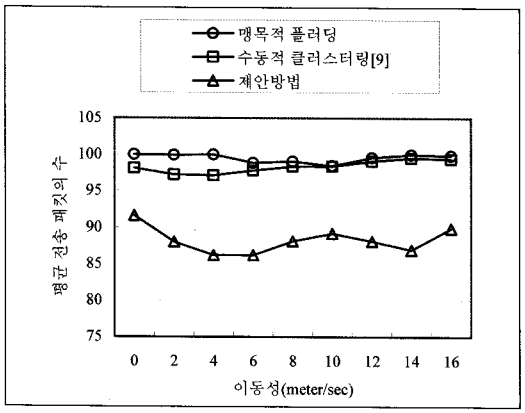


그림 4 단일 플러딩 당 전송된 평균 패킷 수 vs. 이동성

그림 4는 실험영역 1000㎡ 에서 이동속도의 변화에 따른 한번의 플러딩을 완료하기 위해 요구되는 평균 전송 패킷수의 변화를 보여준다. 그림 4에서 보는 것처럼, 제안하는 방법에서 플러딩 효율성은 다른 두 가지 방법에 비해 훨씬 좋다. 이것은 다음과 같은 사실들에 의해 기인한다. 수동적 클러스터링은 게이트웨이 선택 절차를 수행하므로 맹목적 플러딩방법과는 다르게 몇 개의 비 플러딩노드를 생성할 수 있다. 그러나 수동적 클러스터링에서 임의의 노드가 비 플러딩노드가 되기 위해서는 먼저 세가지 상태(게이트웨이 준비, 게이트웨이, 분산 게이트웨이)중 하나에 속해야 하고 최소한 세 개 이상의 이웃 플러딩노드를 알고 있어야 한다. 그러나 방송패킷의 잦은 충돌발생으로 인해 이웃의 플러딩노드의 파악은 쉽지 않다. 따라서 수동적 클러스터링은 제한된 수의 비 플러딩노드 만을 생성할 수 있다. 반면에, 제안하는 방법에서 임의의 노드는 자신과 같은 역할 및 선행자를 가진 노드를 발견하면 비 플러딩노드로 변경될 수 있다. 더구나 제안하는 방법은 유니캐스트 패킷 전송을 통해 그러한 노드들을 가능한 많이 파악할 수 있다. 즉, 비 플러딩노드가 되기 위한 선택에 참여하는 노드의 수가 많아지므로, 비 플러딩 노드의 수도 많아진다.

그림 5는 실험영역 1000 ㎡에서 이동속도의 변화에 따른 한 번의 플러딩동안 패킷을 수신하는 노드의 평균 비율을 보여준다. 그림 5에서 보는 것처럼, 제안하는 방법은 4m/s에서부터 8m/s의 이동속도에서 두 가지 다른 방법들에 비해 낮은 수신율을 보인다. 이는 그림 4에서 보는 것처럼, 4m/s에서부터 8m/s의 이동속도에서 전송되는 패킷의 수가 작기 때문이다. 그러나 다른 노드 이동성을 가진 상황에서 제안하는 방법은 다른 두 가지 방법과 거의 유사한 수신율을 보장한다.

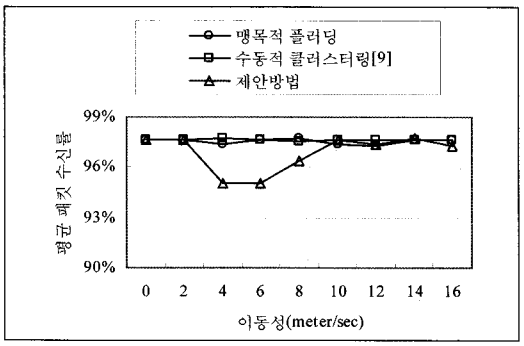


그림 5 단일 플러딩 당 평균 패킷 수신률 vs. 이동성

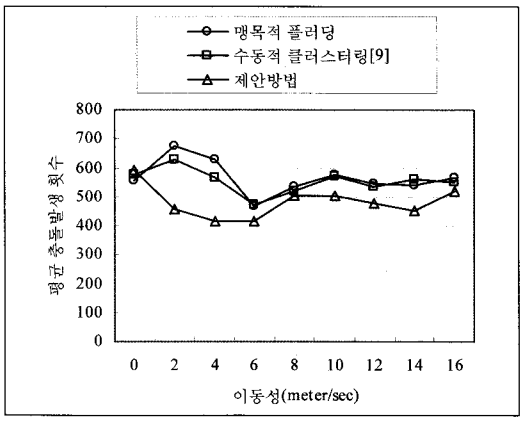


그림 6 단일 플러딩 당 평균 충돌 발생횟수 vs. 이동성

그림 6은 실험영역 1000㎡에서 이동속도의 변화에 따른 한번의 플러딩에 대해 발생하는 평균 충돌횟수를 보여준다. 그림 6에서 보는 것처럼, 제안하는 방법은 다른 두가지 방법에 비해 작은 수의 충돌을 발생시킨다. 이는 제안하는 방법에서 플러딩소스를 제외한 모든 노드들이 유니캐스트 패킷을 전송하기 때문이다. 즉, 유니캐스트 패킷전송은 IEEE 802.11의 MAC계층에서 동작하는 RTS/CTS 핸드쉐이킹에 의해 충돌을 회피하기 때문에, 충돌의 발생횟수를 감소시킨다.

그림 7, 8, 9는 노드들이 이동하지 않는 네트워크에서 실험영역의 증가에 따른 전송패킷의 수, 패킷 수신율, 충돌횟수의 변화를 각각 보여준다.

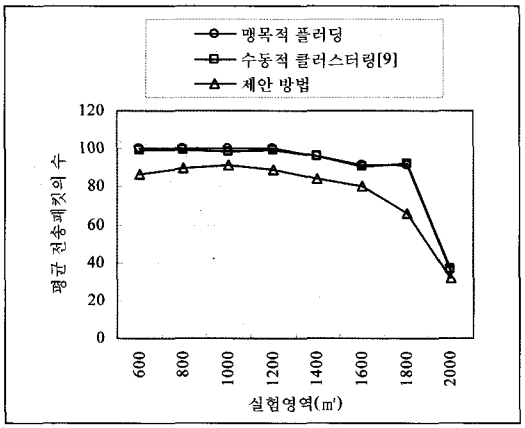


그림 7 단일 플러딩 당 전송된 평균 패킷수 vs. 실험영역

그림 7은 노드들이 이동하지 않는 네트워크에서 실험영역의 변화에 따른 한 번의 플러딩을 완료하기 위해 요구되는 평균 전송 패킷수의 변화를 보여준다. 그림 7에서 보는 것처럼, 실험영역이 증가할수록, 세가지 방법 모두 전송되는 패킷의 수를 감소시킨다. 이는 노드들이 실험영역내에 랜덤하게 배치되기 때문이다. 따라서 실험영역이 증가하면, 노드들간의 연결성은 낮아진다. 즉, 임의의 노드가 새로운 플러딩패킷을 수신하지 않는 동안에는, 전송되는 패킷의 수는 감소한다. 그림 7로부터 알 수있는 또다른 사실은 제안하는 방법은 모든 실험영역에서 다른 두가지 방법에 비해 더 나은 플러딩효율성을 보인다는 것이다. 이는 제안하는 방법이 두 방법에 비해 보다 많은 수의 비 플러딩노드를 생성하기 때문이다.

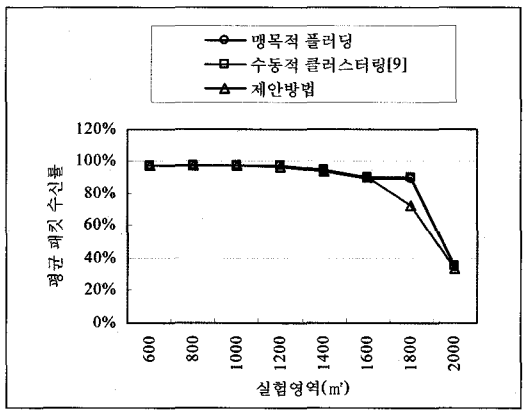


그림 8 단일 플러딩 당 평균 패킷 수신률 vs. 실험영역

그림 8은 노드들이 이동하지 않는 네트워크에서 실험영역의 변화에 따른 한번의 플러딩동안 패킷을 수신하는 노드의 평균 비율을 보여준다. 그림 8에서 보는 것처럼, 실험영역이 증가할수록, 패킷 수신율은 세가지 방법에서 모두 감소한다. 이는 실험영역이 증가함에 따라 노드들간의 연결성이 저하되기 때문이다. 그림 8로부터 알 수 있는 또 다른 사실은 제안하는 방법이 1800㎡의 실험영역을 제외하면 정적인 네트워크(static network)에서는 다른 방법과 거의 유사한 패킷 수신율을 보장한다는 것이다.

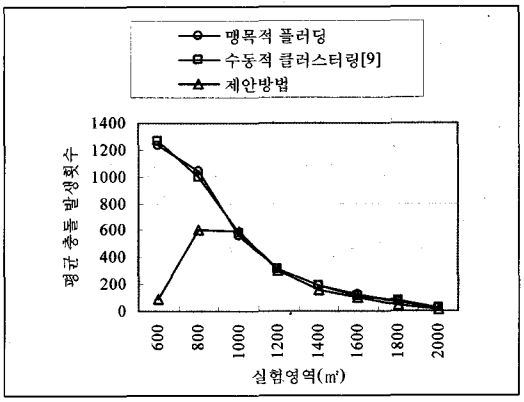


그림 9 단일 플러딩 당 평균 충돌 발생횟수 vs. 실험영역

그림 9는 노드들이 이동하지 않는 네트워크에서 실험영역의 변화에 따른 한번의 플러딩 에 대해 발생하는 평균 충돌횟수를 보여준다. 그림 9에서 보는 것처럼, 맹목적 플러딩과 수동적 클러스터링은 실험영역이 증가함에 따라 충돌 발생 횟수가 크게 감소한다. 만일 실험영역이 좁으면 노드들간의 밀집도가 크게 증가한다. 따라서 높은 밀집도하에서 방송패킷에 의한 충돌횟수는 크게 증가한다. 실험영역이 증가함에 따라 노드들간의 연결성은 크게 감소된다. 따라서 패킷을 방송하기 위해 경합하는 노드의 수도 감소한다. 반면에, 제안하는 방법은 좁은 실험영역(600㎡, 800㎡)에서 다른 두가지 방법에 비해 충돌 횟수를 크게 감소시킨다. 이는 제안하는 방법에서는 각 노드들이 유니캐스트패킷 전송을 이용하여 충돌을 회피하기 때문이다. 실험영역이 증가함에 따라 제안하는 방법의 충돌 감소율도 크게 감소한다. 그럼에도 불구하고, 제안하는 방법은 다른 두가지 방법에 비해 보다 작은 수의 충돌을 유발한다.

4.3 논의

기존에 애드 혹 네트워크에서 플러딩을 효율적으로 수행하기 위한 다양한 방법들이 소개되었지만, 이들은 클러스터 구성과는 상관없이 플러딩의 효율화 만을 꾀하고 있다. 반면에 수동적 클러스터링 기법은 플러딩 패킷을 이용하여 클러스터를 구성 및 유지보수 한다. 이로 인해 수동적 클러스터링 기법은 클러스터의 유지보수에

드는 비용을 크게 감소 시킬 수 있는 이점을 제공한다. 제안된 방법 역시 플러딩을 이용하여 클러스터를 구성 및 유지보수 하므로, 같은 이점을 제공한다. 따라서 제안된 방법은 수동적 클러스터링 방법과 비교되었다. 그러나 수동적 클러스터링이 방송 패킷만을 이용하여 클러스터를 구성하는 반면에 제안하는 방법은 유니캐스트 패킷만을 이용한다. 이로 인해 제안된 방법은 방송 패킷에 의해 자주 발생하는 충돌을 크게 감소시킬 수 있다. 이러한 충돌감소는 보다 많은 노드들이 이웃노드의 수 및 클러스터 상태를 파악할 수 있게 하고 결과적으로 보다 많은 노드들에게 플러딩을 중단할 수 있는 기회를 부여한다.

기존의 요구형 라우팅 프로토콜들에서 임의의 소스노드는 임의의 목적노드로의 경로를 가지고 있지 않을 때 플러딩기법을 사용하여 그 경로를 발견한다. 즉, 경로요청(Route Request) 패킷만 플러딩기법을 사용하여 전파되고 다른 제어패킷 및 데이타 패킷은 유니캐스트 방식으로 전파된다. 따라서 임의의 반응적 라우팅 프로토콜에 각각 다른 플러딩기법들을 적용한다면, 그 프로토콜의 성능에 영향을 미치는 요소들은 경로요청 패킷 전송횟수, 경로요청 패킷 수신률, 경로요청 패킷의 수신지연시간 등이다. 그러나 패킷 전송횟수, 패킷 수신률은 이미 4.2절에서 제공되었으므로, 이 절에서는 수신지연시간에 대한 실험결과만을 제공한다. 표1의 실험 파라미터들과 그 값들을 이용하여 한번의 플러딩에 대한 수신패킷의 평균지연시간이 측정되었다.

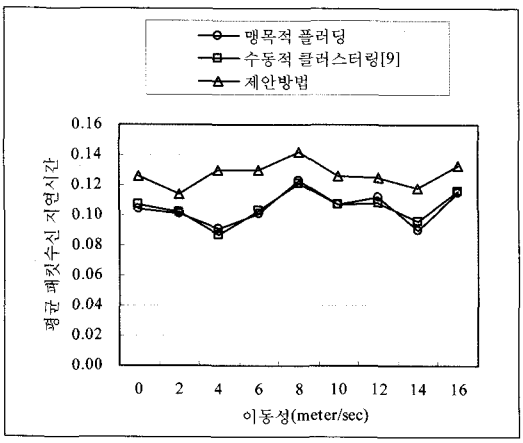


그림 10 단일 플러딩 당 평균 패킷수신 지연시간 vs. 이동성

그림 10은 실험영역 1000㎡ 에서 이동속도의 변화에 따른 한번의 플러딩에 대한 패킷수신을 위한 평균지연시간을 보여준다. 그림 13에서 보는 것처럼, 제안하는 방법은 맹목적 플러딩과 수동적 클러스터링에 비해 패킷수신을 위한 지연시간이 조금 더 길다. 이는 제안하는 방법이 유니캐스트 패킷전송을 하기 때문이다. 즉, 유니캐스트 패킷전송은 충돌회피 절차를 사용하기 때문에 방송 패킷전송에 비해 약간의 지연시간을 더 요구한다.

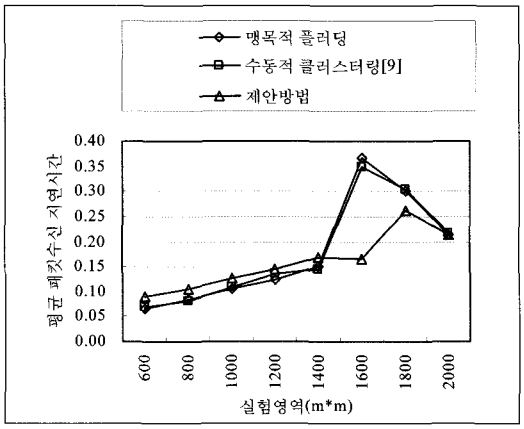


그림 11 단일 플러딩 당 평균 패킷수신 지연시간 vs. 실험영역

그림 11은 노드들이 이동하지 않는 상황에서 실험영역의 변화에 따른 한번의 플러딩에 대한 패킷수신을 위한 평균 지연시간을 보여준다. 그림 11에서 보는 것처럼, 노드들의 밀집도가 높은 경우(<=1400㎡)에 제안하는 방법은 방송패킷 기반 방법들(맹목적 플러딩, 수동적 클러스터링)에 비해 패킷수신을 위한 지연시간이 더 길다. 이는 유니캐스트 패킷의 충돌회피절차로 인한 것이다. 반면에 노드들의 밀집도가 낮은 경우(>1400㎡)에 방송패킷 기반 방법들은 더 긴 수신지연시간을 요구한다. 이는 노드들간의 연결성이 낮은 상황에서 방송패킷의 충돌로 인해 각 노드가 패킷을 우회하여 수신하는 경우가 자주 발생하기 때문이다.

5. 결 론

본 논문에서 제안하는 클러스터 구성 방법은 다음과 같은 이점을 지닌다. 먼저, 제안하는 방법은 주기적인 메시지 교환없이 요구형으로 클러스터를 구성한다. 둘째, 제안하는 방법은 유니캐스트 패킷을 전송하므로 이웃 플러딩노드의 파악이 용이하다. 따라서 제안하는 방법은 이들 이웃 플러딩노드 중에서 일부만을 플러딩노드로 선정함으로써 플러딩노드의 수를 많이 감소시킬 수 있다. 마지막으로, 제안하는 방법은 유니캐스트 패킷 전송으로 인해 충돌의 횟수를 크게 감소시킨다.

동일한 실험영역 하에서 제안하는 방법은 수동적 클러스터링에 비해 약 10%의 패킷전송을 감소시켰고 13%

의 충돌을 감소시켰다. 이동성이 없는 네트워크 하에서 제안하는 방법은 수동적 클러스터링에 비해 약 13%의 패킷전송을 감소시켰다. 또한 제안하는 방법은 수동적 클러스터링에 비해 중간이상의 실험영역(>1000㎡)에서 약 5%의 충돌을 감소시켰다. 향후 연구과제는 제안하는 클러스터 구성 방법을 기존의 반응적 라우팅 프로토콜들에 적용시켜 그 효과를 분석하는 것이다.

참 고 문 헌

[1] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing for Multihop Wireless Ad Hoc Networks," in Ad Hoc Networking, Ed. C. E. Perkins: Addison-Wesley, 2000, pp. 139-168.

[2] M. Jiang, J. Li, Y. C. Tay, "Cluster Based Routing Protocol (CBRP) functional specification," IETF draft 1998.

[3] C. E. Perkins and E. M. Royer, "The Ad Hoc On-Demand Distance-Vector Protocol," in Ad Hoc Networking, Ed. C. E. Perkins: Addison-Wesley, 2000, pp. 173-219.

[4] 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.

[5] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying:An efficient technique for Flooding in mobile wireless networks," Tech. Rep. RR-3898, INRIA-Rocquencourt, Mar. 2000.

[6] M. Seddigh, J. Solano and I. Stojmenovic, "Internal nodes based broadcasting algorithms in wireless networks," in *Proc. of the 34th annual HICSS 2001*, 2001.

[7] H. Lim and C. Kim, "Flooding in Wireless Ad Hoc Networks," in *Proc. ACM MSWiM Workshop at MOBICOM*, Aug. 2000.

[8] K. Mase, Y. Wada, N. Mori, K. Nakano, and M. Sengoku, "Flooding Schemes for Clustered Ad Hoc Networks," *IEICE Trans. Communications*, vol. E85-B, no. 3, pp. 605-613, Mar. 2002.

[9] 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.

[10] M. Gerla and C. Chiang, "Multicluster, Mobile, Multimedia Radio Network," *ACM-Baltzer J. Wireless Networks*, vol. 1, no. 3, pp. 255-265, 1995.

[11] Wireless and Mobility Extensions to the ns-2 Network Simulator. CMU Monarch Project, Available: http://monarch.cs.cmu.edu/cmu-ns.html

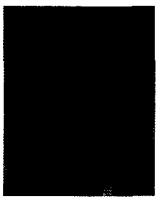
[12] Network Simulator. ns-2. UC Berkeley, Available: http://www.isi.edu/nsnam/ns

[13] J. Yoo, H. Gil, and C. Kim, "INK: Implicit Neighbor Knowledge Routing in Ad Hoc Networks," in *Proc. of IEEE VTC 2003-spring*, Jeju Korea, Apr. 2003, pp. 1826-1830.

[14] T. C. Hou and T. J. Tsai, "An Access-Based Clustering Protocol for Multihop Wireless Ad Hoc Networks," IEEE JSAC, 19(7), pp. 1201-1210, 2001.

왕 기 철

1997년 광주대학교 전자계산학과 졸업(공학사). 2000년 목포대학교 멀티미디어공학과 졸업(공학석사). 2005년 전북대학교 컴퓨터통계정보학과 졸업(이학박사) 현재 전북대학교 영상정보신기술연구소(CAIIT) 연구원. 관심분야는 이동컴퓨팅, Ad hoc 네트워크, 무선네트워크 보안



김 태 연

1985년 전남대학교 계산통계학과(이학사) 1988년 전남대학교 대학원 전산통계학과(이학석사). 1996년 전남대학교 대학원 전산통계학과(이학박사). 1996년~현재 서남대학교 컴퓨터정보통신학과 조교수 관심분야는 네트워크 보안, 이동 컴퓨팅, 네트워크 관리

조 기 환

1985년 전남대학교 계산통계학과 졸업(학사). 1987년 서울대학교 계산통계학과 졸업(석사). 1996년 영국 Newcastle 대학교 전산학과 졸업(박사). 1987년~1997년 한국전자통신연구원 선임연구원. 1997년~1999년 목포대학교 컴퓨터과학과 전임강사. 1999년~현재 전북대학교 전자정보공학부 부교수 관심분야는 이동컴퓨팅, 컴퓨터통신, 무선네트워크 보안, 센서네트워크, 분산처리시스템