모바일 애드 혹 네트워크에서의 주소 예약을 이용한 IP 주소 자동 설정 기법

(An IP-address Auto-configuration Technique using Address Reservation for a Mobile Ad hoc Networks)

김 남 훈 † 안 소 연 † 문 경 덕 †† 이 영 희 †††
(Namhoon Kim) (Soyeon Ahn) (Kyeongdeok Moon) (Younghee Lee)

요 약 모바일 애드 혹 네트워크(MANET, Mobile Ad hoc Networks)란 중앙 집중 형태의 제어나 인프라 구조가 없는 독립적인 모바일 컴퓨팅 노드들간의 집합을 의미한다. 애드 혹 네트워크에 연결을 원하는 모든 노드들은 네트워크 연결에 필요한 주소를 얻어야 한다. 만일 DHCP(Dynamic Host Configuration Protocol)와 같이 자동으로 주소를 할당해 주는 서버가 존재한다면 쉽게 주소를 얻을 수 있지만, 모바일 애드 혹 네트워크의 특성상 DHCP와 같은 서버의 존재를 가정하기 어렵다. 본 논문에서는 주소 예약(Address Reservation)과 낙천적 중복 주소 검색(Optimistic Duplicated Address Detection)을 이용한 분산형 자동 주소 설정 방법을 제시한다. 주소 예약은 주소 할당 시간을 줄이는 데 도움을 주고, 낙천적 DAD는 주소의 유일성을 보장하고 네트워크의 트래픽 오버헤드를 줄이는 데 기여한다. 또한 제한된 방법은 모바일 애드 혹 네트워크의 분할과 합병에 대한 해결책을 제시한다. 마지막으로 ns-시뮬레이터를 사용한 모의 실험의 결과를 통해 제안된 방법이 주소 유일성 보장, 주소 설정 시간과 통신 오버헤드 측면에 있어서 우수함을 증명한다.

키워드 : 자동 설정, 주소 예약, 낙천적 중복 주소 검색, 모바일 애드 혹 네트워크

Abstract A Mobile Ad hoc Network (MANET) is a group of independent mobile computing nodes that consist of a multi-hop wireless network without a central administration or any infrastructure. Every node that wants to join a MANET must obtain an address for communication. Having a centralized DHCP server that provides addresses to nodes, we can easily and automatically obtain addresses. However, a MANET lacks any fixed infrastructure such as a DHCP server. We therefore propose a distributed address autoconfiguration approach for a MANET using a reserved address and optimistic Duplicated Address Detection (DAD). The reserved address helps to reduce the allocation latency, and the optimistic DAD guarantees the uniqueness of addresses and lessens communication overhead. We then suggest methods of handling network partition and network merging situations, and go on to evaluate our approach through simulations. The simulation result shows that our scheme guarantees the uniqueness of allocated address and considerably improves allocation latency and communication overheads.

Key words : autoconfiguration, address reservation, optimistic DAD, ad hoc network

· 이 논문은 정보통신연구진흥원(A1100-0300-0004)과 한국과학재단의 특성 기초 사업(R01-2003-000-10562-0)으로부터 지원받았음
† 비 회 원 : 한국성보통신대학교 공학부
nhkim@icu.ac.kr
syahn@icu.ac.kr
†† 비 회 원 : 한국전자통신연구원 홈네트워크미들웨어연구팀 팀장
kdmoon@etri.re.kr
††† 종신회원 : 한국성보통신대학교 공학부 교수
yhlee@icu.ac.kr
논문접수 : 2004년 5월 19일
심사완료 : 2004년 8월 18일

1. 서 론

네트워킹 기능과 이동 장비가 일상 생활에서 차지 하는 비중이 점차 커지고 그 저변이 확대됨에 따라 모바일 애드 혹 네트워크 또한 주목을 끌어왔다. 모바일 애드 혹 네트워크란 한 그룹의 모바일 무선 장비들로 구성되는 임의의 네트워크로써, 어떤 고정된 인프라 구조나 관리가 존재하지 않고, 노드들의 움직임으로 인해 네트워크 토폴로지가 매우 자주 변하는 네트워크를 의미한다. 이러한 모바일 애드 혹 네트워크의 특성들은 많은

연구 주제들을 제공해 왔다.

모든 노드들이 서로 통신을 하기 위해서 네트워크 인터페이스의 주소 설정은 가장 기본이 되고 필요한 절차임에도 불구하고, 기존의 모바일 애드 혹 네트워크 연구들에서 라우팅 프로토콜에서는 이미 모든 노드들은 각자의 주소를 가지고 있다고 가정한 상태에서 연구하였다[1-3].

유선 네트워크 환경에서는 DHCP 서버를 이용하여 IP 주소, 서브넷 주소, 기본 게이트웨이 등의 네트워크 설정을 쉽게 할 수 있다[4]. 그러나, 일반적인 모바일 애드 혹 네트워크에서는 DHCP 서버와 같은 고정된 인프라 구조를 가지고 있지 않기 때문에, 모바일 애드 혹 네트워크를 위한 주소 설정 방법이 요구된다.

모바일 애드 혹 네트워크를 위한 노드의 주소 자동 설정 관련 연구 방향 중에 하나는 한 이동 노드가 DHCP 서버 역할을 하게 하자는 연구들이었다. 이를 중앙 집중형 주소 설정 방법이라 한다. 서버 노드는 주소 풀(pool)을 관리하고 주소 설정에 대한 모든 권한을 갖는다. 즉, 새로운 노드가 네트워크에 들어 왔을 때, 새로운 노드는 주소를 할당 받기 위해 반드시 서버 노드를 찾아야 한다. 이 방법은 매우 간단하지만, 서버 노드에 오버 헤드가 집중 되어 장애가 발생할 수 있는 지점이 된다. 또한, 새로운 노드는 처음에 네트워크 주소가 없으므로, 서버 노드와의 통신에 어려움을 겪을 수 있다[5-7].

다른 한편, 분산형 주소 설정 기법을 제안한 연구들도 있다. 분산형 방법은 하나의 노드가 장애 지점이 되는 경우가 없는 대신, 주소 풀을 관리하고 할당된 주소가 유일하다는 것을 보장하는 데 어려움을 겪는다. 게다가 네트워크가 분리되거나 합쳐질 경우에 해결 방법이 상대적으로 더 복잡하다[8-10].

기존 자동 주소 설정 방법들이 위에서 제시한 문제점들을 보여줌으로써, 효과적인 자동 주소 설정 방법은 무엇이고 어떻게 그것을 설계할 수 있는가 하는 질문을 야기한다. 본 논문에서는 자동 주소 설정 방법이 갖춰야 할 주요한 요구 사항들을 다음과 같이 정리하였다.

첫째, 무엇보다도 할당된 주소의 유일성을 보장해야 한다. 모든 모바일 애드 혹 네트워크의 노드들은 올바르게 동작하기 위해 유일한 주소를 요구한다. 왜냐하면, 주소간의 충돌이 있으면, 노드들 간의 잘못된 정보를 전달하거나 저장하게 된다. 이런 잘못된 정보들은 결국 올바르지 못한 라우팅 혹은 응용 프로그램의 오 동작 등을 초래하게 된다. 주소 충돌은 전체적인 네트워크 성능에도 영향을 미칠 수 있으므로, 할당 주소에 대한 유일성 보장은 무엇보다도 중요하다고 할 수 있다.

둘째, 새로운 노드가 모바일 애드 혹 네트워크에 빨리 들어올 수 있도록, 주소 설정 시간은 가능한 작아야 한다. 기존의 방법들 중에서 모바일 애드 혹 네트워크의 홉 수에 영향을 받는 방법들도 있기 때문에 이 또한 중요한 고려 사항이 된다.

셋째, 주소 설정 방법은 전체 노드 수와 네트워크의 크기에 영향을 받지 않아야 한다. 네트워크 사이즈와 노드 수가 많아지면 자연히 많은 통신 오버헤드를 요구하게 되지만, 효율적인 주소 설정 방법은 가능한 이러한 통신 오버헤드를 줄여야 한다는 것이다.

마지막으로, 주소 설정 방법은 주소 낭비, 네트워크의 분리와 통합 문제 그리고 메시지 분실에 대한 문제들을 고려해야 한다. 다시 말해, 좋은 주소 설정 방법은 네트워크에 상황에 무관하게 정상적으로 운영될 수 있어야 한다는 것이다.

기존의 주소 자동 설정 방법들은 위에 제시된 요구사항을 모두 충족시키는 데 제한을 갖고 있다. 따라서, 본 논문에서는 모바일 애드 혹 네트워크 환경에서 제시된 요구 사항을 만족하는 분산형 주소 설정 방법을 제안한다. 제안된 방법은 예약 주소를 이용하여 새로 네트워크에 들어오는 노드에게 빠르게 주소를 할당해 준다. 이것은 DHCP 서버의 개념과 유사하지만, 중앙 집중형 방법은 아니다. 또한 제안된 방법은 할당될 주소의 유일성을 보장하기 위해 사용되는 DAD 회수를 줄이기 위해 낙천적 DAD를 사용한다. 그리고 다양한 네트워크 환경에서도 동작할 수 있도록 노드의 고장과 이동, 네트워크의 분리와 통합 등에 대해서도 다룬다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 주소 자동 설정 방법들을 기술하고, 3장에서는 제안된 주소 자동 설정 방법의 기본적 동작에 대해서 기술한다. 4장에서는 비정상적인 환경에서의 제안된 방법이 어떻게 동작하는 가를 설명한다. 5장에서는 제안된 방법을 기존의 방법들과 비교해 그 성능을 평가해 보고, 마지막으로 6장에서 본 연구의 결론과 향후 연구 방향에 대해서 기술한다.

2. 관련 연구

2.1 중앙 집중형 방법들(The Centralized Schemes)

중앙 집중형 방법들은 각각의 모바일 애드 혹 네트워크를 위한 서버 노드를 선출해야만 한다. [5],[6],[7]에서 중앙 집중형 방법들의 예를 볼 수 있다. 서버 노드들의 이름은 각각의 방법에 따라 조금씩 다르다(agent node [5], leader node [6] or address authority [7]). 그러나 서버 노드의 역할은 매우 유사하다. 즉, 서버 노드는 주소 풀 혹은 리스트를 관리하고 주소 할당에 관한 책임과 권한을 가진다. 그리고 두 개 이상의 모바일 애드 혹 네트워크들이 통합되면, 서버 노드의 숫자가 두 개 이상

이 될 수 있는데, 이 때 반드시 서버 노드의 수를 하나로 줄여야 한다.

중앙 집중형 방법들은 서버 노드들이 주소 풀을 관리하기 때문에 여러 네트워크의 통합에 의한 주소 충돌 문제를 매우 간단히 해결할 수 있다. 그러나, 서버 노드에 많은 부담이 가중되고, 서버 노드가 장애 발생 지점이 될 수 있다. 게다가, 네트워크의 홉이 늘어나게 되면, 이 방법들은 서버 노드를 찾는데 시간이 증가되고 주소를 할당 받는 시간 또한 증가될 것이다.

2.2 분산형 방법들(The Distributed Schemes)

주소 요청(Address Request, AREQ) 과 주소 응답(Address Reply, AREP) 메시지를 이용한 분산 주소 자동 설정 방법인 강력한 중복 주소 검색(Strong Duplicated Address Detection, 이하 Strong DAD) 방법이 제안되었다[8]. 이 방법은 IETF Zeroconf working 그룹의 기본 방향과 의견을 일치하고 있다. 주소를 할당 받기를 원하는 새로운 노드는 자신이 사용할 주소를 임의로 선택하고 그 주소가 현재 모바일 애드 혹 네트워크에서 사용되고 있는지를 검사한다(DAD). 이 때, 새로운 노드는 기존의 네트워크와 통신하기 위해 임시 주소를 임의로 선택하여 사용한다. 그러나, 이 임시 주소의 충돌은 확인하지 않고 사용하는 문제를 가지고 있다. 게다가, 주소를 얻을 때까지 계속 DAD 과정을 반복하므로, 주소 설정 시간이 DAD 과정의 실패 회수에 비례한다. 네트워크의 홉 수 또한 할당 시간에 영향을 줄 수도 있다. 뿐만 아니라, 이 방법은 네트워크 분리와 통합에 대해 고려하지 않고 있다.

Sanket과 Ravi는 분산 동의(distributed agreement) 개념을 이용하여 에이전트 기반 분산 주소 자동 설정 방법(이하 MANETconf)을 제안하였다[9]. MANETconf에서는 Strong DAD와 달리, 주소 설정을 위해 새로운 노드가 기존의 모바일 애드 혹 네트워크 노드 중 하나를 자신의 에이전트로 선택한다. 이 때, 새로운 노드를 requestor, 에이전트 노드를 initiator라 칭한다. 에이전트 노드는 하나의 주소를 임의로 선택하고 다른 모든 네트워크 노드들에게 그 주소의 사용에 대한 동의를 얻은 후 새로운 노드에게 그 주소를 할당한다. 이 방법에서는 동의를 얻기 위한 과정에서, 에이전트 노드가 NACK을 받는 것이 아니라 ACK을 받는 수정된 DAD를 사용한다. 이러한 방법은 많은 ACK이 한꺼번에 몰리는 ACK implosion 현상을 만들어 낼 수 있다는 문제점이 있다. 에이전트 노드는 Timeout까지 기다리지 않고 메시지를 받아야 하는 노드들의 리스트를 관리한다. 네트워크 분리와 통합 문제를 위한 해결책으로 가장 낮은 주소를 가진 노드를 그룹의 리더로 정하고, 그룹의 리더가 주기적으로 UUID(Universally Unique Identifier)란 일종의 그룹 ID를 포함한 메시지를 네트워크 전체에 전송 한다. 만일 하나의 노드가 리더 노드로부터 메시지를 받지 못하면, 그 노드는 네트워크 분리를 감지한다. 반대로, 한 노드가 두 개 이상의 리더 노드로부터 메시지를 받게 되면, 그 노드는 네트워크의 통합을 감지한다. MANETconf는 모든 노드의 리스트를 관리하고 각 노드로부터의 응답 메시지를 수집하기 때문에, 다른 노드들로부터의 동의를 얻는데 성공했는지 실패했는지를 Strong DAD보다 빨리 결정할 수 있다. 또한, 모든 노드들의 리스트를 관리하는 것은 오버헤드가 될 수 있지만, 네트워크가 통합되었을 때 발생되는 주소 충돌 문제 해결에 도움이 된다. 그러나, 이 방법 또한 에이전트 노드가 동의를 얻는데 실패하는 것에 비례하여 주소 할당 시간이 길어지고, ACK을 사용하기 때문에 전체적인 통신 오버헤드는 별다른 차이가 없다.

Hongbo와 그의 동료들은 정수 수열을 생산하는 함수를 이용한, 주소 충돌이 없는 분산형 자동 주소 설정 방법을(이하 Prophet) 제안하였다[10]. 모바일 애드 혹 네트워크에서의 주소는 노드가 네트워크에 머무는 동안 충돌이 없어야만 한다. 따라서, 저자들은 똑 같은 숫자가 생성 되는데 아주 오랜 시간이 걸리는 특성을 만족하는 함수를 고안하려고 했다. 또한, 그 함수는 임의의 시간 간격에서 서로 다른 seed를 가지는 상황에서 같은 숫자를 생성하는 확률도 아주 작아야 한다. 그러나 분산 환경에서 이러한 함수를 고안하는 것은 매우 어려운 일이다. Prophet은 주소 할당 시간과 통신 오버헤드 측면에서 명백한 장점을 가진다. 그러나, 이 방법은 저자들이 주장하는 것과는 달리 주소 충돌 확률이 매우 높다는 치명적인 약점을 가진다. 주소 길이가 16비트일 때, 주소 충돌 확률은 모바일 애드 혹 네트워크의 자동 주소 설정 방법으로 사용하기에 매우 높다. 물론 저자들이 제안한 주소 길이인 24 bit를 사용하고, 네트워크의 노드 수 또한 150보다 작게 한다면 주소 충돌은 확률은 매우 작은 값이다. 그러나, 만일 주소 길이가 24bit이고 전체 노드 수가 150을 넘지 않는다면, 임의로 주소를 선택해서 써도 주소 충돌 확률은 Prophet과 마찬가지로 거의 0에 가깝다. 게다가 전체 사용할 수 있는 주소의 수는 무한대가 아니기 때문에 충돌은 항상 존재할 수 있다. 그래서 Prophet은 주소 충돌을 피하기 위해 명시적인 DAD를 거쳐야 한다.

이 밖에도, 주소 충돌을 피하기 위해 계속 주소 영역을 자르는 분산형 방법들이 존재한다. 그러나, 이런 방법들은 안 쓰거나 못 쓰는 주소가 없도록 하기 위해 추가적으로 주소를 재 수집하는 방법이 요구된다[11-13]. 그리고 중앙 집중형 방식과 분산형 방식의 중간 형태인 방법들을 [14], [15]에서 볼 수 있다. 이 방법들은 서브

네트워크 개념을 이용하였다. 또, 어떤 연구자들은 주소 설정 방법을 다른 관점으로 연구한다. 그들은 직접적인 설정 방법을 제안하는 것이 아니라, 라우팅 프로토콜의 도움으로 주소 충돌을 발견하고 이를 해결하는 방법을 제시하였다[16,17]. 그 외 특이한 방법으로, [18]에서는 새로운 노드가 항상 리더 노드가 되고 주소 또한 항상 1씩 증가하는 방법을 제안하였고, [19]는 IP 주소 대신 가변적 주소 길이를 가지는 형태로 주소를 설정하는 방법을 제안하였다.

3. 예약 주소를 이용한 주소 자동 설정 방법

DHCP 서버를 대신할 기존의 자동 주소 설정에 관한 연구들을 살펴본 결과 기존의 방법들이 본 논문에서 제시한 자동 주소 설정이 가져야 할 기본적인 요구 사항을 모두 충족하지 못하며, 또한 각 방법들마다 단점을 가지고 있는 것을 알았다.

본 논문에서는 주소 자동 설정 방법의 요구 사항을 만족하고 기존의 방법들의 단점을 극복하기 위해, 예약 주소와 낙천적 DAD 기법을 이용한 분산형 주소 자동 설정 방법을 제안한다.

예약 주소를 가진 모바일 애드 혹 네트워크 노드는 새로운 노드에게 빨리 주소를 할당 해 줄 수 있다. 다시 말해, 일부 모바일 애드 혹 네트워크 노드들은 작은 DHCP 서버 역할을 한다고 볼 수 있다. 만일 새로운 노드가 네트워크에 접근하여, 주소를 요청했을 때, 기존 노드들은 자신이 가지고 있는 예약 주소를 즉시 새로운 노드에게 제공할 수 있다. 우리는 이런 역할을 하는 노드를 에이전트 노드라 부르겠다.

에이전트 노드는 새로운 노드에게 주소를 할당 한 후, 낙천적 DAD를 사용해 또 다른 새로운 노드를 위한 주소를 미리 준비한다. 이 때, 에이전트 노드는 기존의 다른 노드들이 사용하고 있지 않는 2개의 주소를 찾는다. 이를 통해 얻은 주소들은 에이전트 노드 자신을 위한 예약 주소와 자신이 주소를 할당해 준 새 노드의 예약 주소로 사용된다. 이런 과정이 끝나면, 에이전트 노드는 하나의 예약 주소를 가지는 모바일 애드 혹 네트워크 노드가 되어 또 다른 새로운 노드가 주소를 요청했을 때 즉시 주소를 줄 수 있게 된다.

본 논문에서는 세가지 형태의 모바일 애드 혹 네트워크 노드를 정의한다. 1) 예약 주소를 가진 모바일 애드 혹 네트워크 노드, 2) 예약 주소를 가지고 있지 않는 모바일 애드 혹 네트워크 노드, 3) 새로운 노드로 정의 한다. 만일 에이전트 노드가 다른 노드들의 부정적 응답 없이 새로운 예약 주소를 구한다면, 에이전트 노드는 1번 형태의 노드가 된다. 그러나, 에이전트 노드가 예약 주소를 구하는 데 실패 한다면, 에이전트 노드는 예약 주소가 없는 2번째 형태의 노드가 된다. 이 때, DAD 과정은 처음 선택된 임의 주소에 대해서 오직 한번만 수행한다. 즉, Strong DAD나 MANETconf와 달리, 만일 임의로 선택한 주소가 이미 사용되고 있어서 다른 노드로부터 NACK을 받게 되면, DAD 과정을 다시 하지 않는다. 왜냐하면, 모든 모바일 애드 혹 노드들이 모두 1번 형태일 필요는 없기 때문이다. DAD의 목적은 다른 노드들이 사용하지 않는 주소를 찾아내는 것인데, 제안된 방법은 그것이 성공하던 성공하지 않던 상관없다. 따라서 제안된 방법에서 사용되는 DAD를 낙천적 DAD라 부른다. 2번 형태의 모바일 애드 혹 네트워크 노드는 새로운 노드에 의해 에이전트 노드로 선택될 때, 1번 형태의 노드가 될 기회를 갖는다.

한편, 새로운 노드가 2번 형태의 노드를 에이전트 노드로 선택하면 즉시 주소를 할당 받을 수 없다. 이런 경우를 대비해 제안된 방법은 주소 차용 (borrowing) 방법을 제안하였다. 즉, 에이전트 노드로 선택 받은 2번 형태의 노드는 1번 형태의 노드로부터 주소를 빌려올 수 있다.

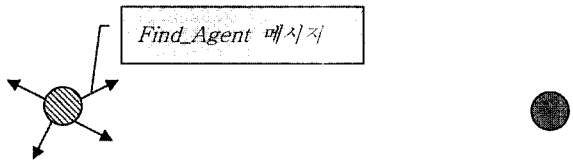
3.1 네트워크 초기화

하나의 노드가 모바일 애드 혹 네트워크에 접근할 때, 노드는 주소를 할당 받기 위해 에이전트 노드를 찾아야 한다(*Find_Agent* 메시지). 만일 새로운 노드가 타임 아웃 전에 응답을(*Reply_Agent* 메시지) 받으면 에이전트 노드를 설정하고 그 에이전트 노드로부터 주소를 구한다. 그러나 응답을 받지 못하면, 새로운 노드는 스스로 주소를 설정하고 또한 스스로 다음 노드를 위해 예약 주소를 준비해 둔다. 이러한 과정이 끝나게 되면 모바일 애드 혹 네트워크의 초기화 과정은 완료된 것이다. 그림 1은 모바일 애드 혹 네트워크의 초기화 과정을 보여준다. 만일 하나 혹은 그 이상의 노드가 동시에 네트워크 초기화를 시도하게 된다면, 각각의 최초 노드는 하나의 모바일 애드 혹 네트워크를 구성하고 추후 각 네트워크가 통합되는 과정을 거친다.

3.2 주소 설정 과정

제안된 주소 설정 과정은 두 부분으로 분리될 수 있다. 첫 째는 할당 단계, 둘째는 예약 단계이다.

할당 단계: 최초 노드가 아닌 새로운 노드 i가 있고, 노드 i 주위에 여러 모바일 애드 혹 노드들이 존재한다고 가정해 보자. 우선, 노드 i가 한 홉 내에서 자신의 에이전트 노드를 찾는 메시지를 보내고(*Find_Agent* 메시지), 타임 아웃이 발생하기 전에 한 개 이상의 응답을 (*Reply_Agent* 메시지) 받았다고 하자. 노드 i는 주소를 할당 받기 위해 하나의 노드를 선택해 자신의 에이전트 노드로 삼는다. 여기서는 j를 선택했다고 가정한다. 이러한 과정 후 노드 i가 주소를 요청하면(*Get_ADDR* 메시



(a) 최초의 새로운 노드가 에이전트 노드를 찾는다.

(b) 최초의 새로운 노드는 스스로 설정을 하고, 모바일 애드 혹 노드가 된다.

● 예약 주소를 가진 모바일 애드 혹 네트워크 노드 (type 1)

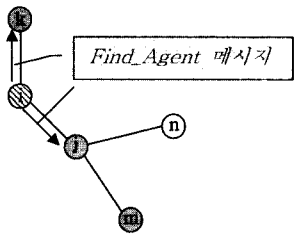
○ 예약 주소가 없는 모바일 애드 혹 네트워크 노드 (type 2)

◎ 새로운 모바일 애드 혹 네트워크 노드 (type 3)

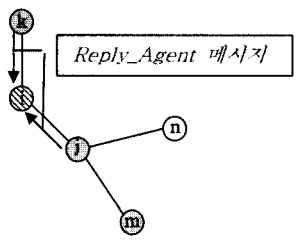
그림 1 네트워크 초기화 과정

지), 노드 j는 즉시 자신의 예약 주소를 노드 i에게 준다 (*Allocation_ADDR* 메시지).

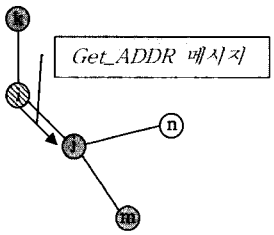
그림 2는 모바일 애드 혹 네트워크 노드가 새로운 노드에게 주소를 주는 할당 단계를 보여준다. 새로운 노드는 예약 주소까지 얻은 후에, 1번 형태의 모바일 애드 혹 네트워크 노드가 된다. 새로운 노드는 일정 시간을 기다린 후에 에이전트가 되겠다는 노드들 중에서 가능하면 예약 주소를 가진 노드를 선택한다(에이전트 노드는 *Reply_Agent* 메시지를 보낼 때 예약 주소를 가지고 있는지를 표시한다). 만일 예약 주소를 가지고 있지 않은 노드를 에이전트로 선택하게 되면, 새로운 노드가 에이전트로부터 바로 주소를 할당 받을 수 없고, 에이전트 노드가 주소를 빌려오는 메커니즘을 통해 주소를 가져와야 하기 때문에 할당 시간이 상대적으로 길어질 수 있다. 그러나, 새로운 노드가 일정 시간을 기다리면, 주변 노드들로부터 예약 주소를 가진 에이전트들로부터 한 개 이상의 응답을 받을 가능성이 높아지므로, 보다 빨리 주소를 할당 받을 수 있다. 즉, 새로운 노드는 그의 에이전트 노드로 2번 형태 보다 1번 형태의 애드 혹 노드를 선택하는 것이 유리하다. 그러나, 얼마나 기다려야 하는지를 결정하기 어렵고, 주위의 모든 노드들의 형태가 2번 형태일 경우엔 기다리는 시간 자체가 낭비가



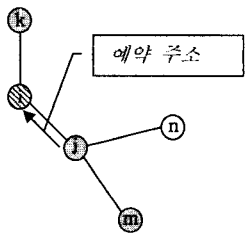
(a) 새로운 노드 i는 에이전트 노드를 찾기 위해 메시지를 보낸다.



(b) 노드 j와 k는 i에게 응답 메시지를 보낸다.



(c) 노드 i는 노드 j를 자신의 에이전트 노드로 선택하고, 주소를 구한다는 메시지를 보낸다.



(d) 노드 j는 자신의 예약 주소를 i에게 할당한다.

그림 2 주소 할당 단계

될 수 있다.

예약 단계: 에이전트 노드 j는 자신과 노드 i를 위한 예약 주소를 준비하기 위해 임의의 주소 2개를 선택해 그것들을 포함한 메시지를 브로드캐스트 한다(*Find_ADDR* 메시지). 만일 타입 1 형태인 노드 k가 해당 메시지를 받으면, 메시지에 포함된 2개의 주소들이 k의 주소, 그리고 k의 예약 주소와 충돌하는 지를 확인한다. 만일 노드 k가 충돌된 주소를 발견한다면, 중복 주소에 대해 부정적 메시지(*NACK_Find_ADDR* 메시지)를 j에게 보내야 한다. 그러나 노드 j가 부정적 응답 메시지 없이 사용 가능한 주소들을 찾는 다면, 노드 j는 두 개의 주소 중 하나를 노드 i에게 준다(*Give_Reserved_ADDR* 메시지). 이 주소는 노드 i의 예약 주소로 사용하게 된다.

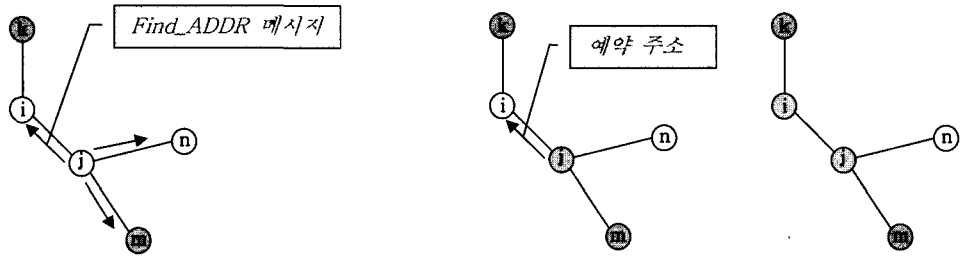
만일 노드 j가 노드 i의 예약 주소를 얻는 데 실패하면, 빈 주소를 담은 부정적 메시지를 i에게 보낸다 (*NACK_Reserved_ADDR* 메시지). 이런 경우 노드 i는 타입 2 형태의 노드가 된다. 물론 노드 j 자신의 예약 주소를 얻는 데도 실패하면, 노드 j 또한 타입 2 형태의 노드가 된다. 이 때, 주소를 다시 찾는 과정은 하지 않는다 (낙천적 DAD). 즉, 낙천적 DAD은 오직 새로운 노드가 모바일 애드 혹 네트워크에 들어올 때만 일어난다. 이제 예약 단계는 완료되었다.

그림 3은 예약 단계를 보여준다. 노드 j가 노드 i에게 예약 주소를 할당한 후에, 노드 i와 j는 타입 2 형태로 변하게 된다. 그러므로 노드 j는 자신과 node i를 위한 2개의 예약 주소를 구할 것이다. 노드 j가 두 개의 주소를 구한 후에 j는 하나를 자신의 예약 주소로 사용하고 다른 하는 노드 i에게 준다. 마지막으로 노드 i와 노드 j는 1번 형태의 모바일 애드 혹 네트워크 노드가 된다.

3.3 예약 주소가 없는 에이전트 노드를 위한 주소 차용 기법

새로운 노드의 에이전트 노드인 모바일 애드 혹 네트워크 노드 j가 예약 주소를 가지고 있지 않다고 가정할 수 있다. 예를 들면, 노드 j가 노드 i에게 주소를 할당한 후, 노드 j는 다음 주소 할당을 위해 낙천적 DAD를 수행한다. 그런데 만일 노드 j가 예약 주소를 얻는 데 실패하고, 새로운 노드 x에 의해 에이전트 노드로 선택받았다면, 노드 x는 j가 예약 주소를 얻는 동안 기다려야 하는가? 만일 노드 x가 노드 j의 과정을 기다린다면, Strong DAD[8] 혹은 MANETconf[9]와 같이 주소를 얻는 데 오랜 시간이 걸릴 것이다. 할당 시간을 줄이기 위해 DAD를 통해 주소를 구하는 대신에 노드 j는 1번 형태의 모바일 애드 혹 네트워크 노드들로부터 예약 주소를 빌려올 수 있다. 그림 4에서 보여주는 것처럼, 만일 노드 j가 차용 메시지를 플러딩 하면 노드 m은 노드 j로부터 주소 차용 요구를 받게 된다(*Borrow_ADDR* 메시지). 그 후, 요청을 받은 1번 형태의 모바일 애드 혹 네트워크 노드들은 노드 j에게 그들의 예약 주소를 빌려줄 수 있다(*Loan_ADDR* 메시지). 그러므로, 노드 j는 많은 응답을 받을 수 있다. 그러나, 노드 j는 오직 하나의 응답만 있으면 된다. 만일 차용 요청에 대한 응답을 받아 전달하는 중간 노드들은 처음 응답만을 전달한다면, 차용 과정 중에 발생하는 메시지의 수를 줄일 수 있을 것이다.

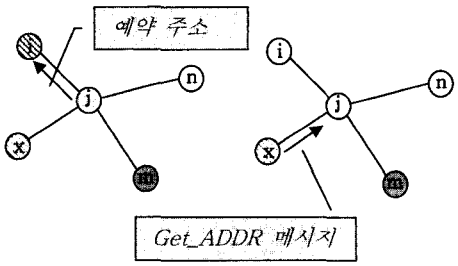
노드 j가 노드 m으로부터 예약 주소를 차용한 후, 노드 j는 노드 x에게 주소를 할당한다. 그리고 노드 j는 노드 m에 승인 메시지를 보낸다(*ACK_Loan_ADDR* 메시지). 마지막으로, 노드 j는 예약 주소를 얻기 위해 낙천적 DAD를 수행한다. 이 경우, 노드 j는 자신과 노드 x, 노드 m의 예약 주소를 위한 3개의 주소들을 임의로 선택해서 이것을 포함한 메시지를 브로드캐스트 한다. 노드 m은 노드 j의 DAD 과정을 끝날 때까지나 j로부터 빈 주소나 예약 주소를 가진 메시지를 받을 때까지는 DAD를 수행하지 않는다. 최악의 경우, 노드 j는 어떤 노드로부터 주소를 빌려 올 수 없을 수 있다. 이



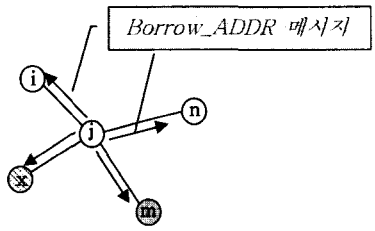
(a) 노드 j는 임의의 두 개 주소를 선택하고 충돌을 확인하기 위해 메시지를 보낸다.

(b) 노드 j는 주소를 얻은 후, 그 중 한 개를 i에게 준다. 마지막으로 노드 i와 j는 2번 형태의 노드가 된다.

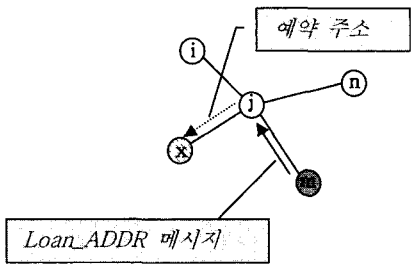
그림 3 주소 예약 단계



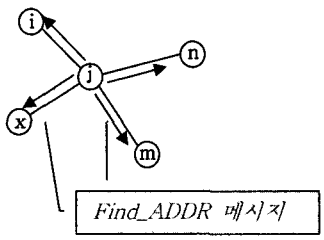
(a) 노드 j는 노드 i에게 주소를 할당한 후, 노드 x는 노드 j에게 주소 할당을 요구한다.



(b) 노드 j는 주소를 차용하기 위해 메시지를 브로드캐스트 한다.



(c) 노드 m은 노드 j에게 자신의 예약 주소를 빌려주고, 노드 j는 이를 이용해 노드 x에게 할당한다.



(d) 노드 j는 세 개의 예약 주소를 얻기 위해 메시지를 플러딩한다.

그림 4 주소 차용 과정

경우 노드 j는 타이머가 종료될 때까지 기다린다. 그 후 노드 j는 자신과 노드 x를 위해 낙천적 DAD를 수행해야 한다. 만일 낙천적 DAD 또한 실패한다면, 노드 x에 대한 주소 할당을 포기한다.

만일 두 개 이상의 노드들이 차용 요청에 대해 응답을 한다면, 그들이 보낸 예약 주소들은 에이전트로부터 승인을 받거나 타이머가 종료될 때까지 다른 노드에게 주거나 혹은 할당되지 않도록 해야 한다. 결과적으로, 가능한 예약 주소들의 수는 주소 차용 과정 중에 줄어들게 된다. 이러한 문제를 막기 위해서 애니캐스트 라우팅 프로토콜이 사용될 수 있다. 그러나 모바일 애드 혹 네트워크에 애니캐스트 라우팅 프로토콜이 존재한다고 가정할 수 없기 때문에, 본 논문에서는 애니캐스트 라우팅이 없는 주소 차용 기법을 제안한다.

전체 DAD 수를 감소시키기 위해서 다음과 같은 방법을 고려해 볼 수 있다. 모든 노드들은 하나 이상의 예약 주소 슬롯을 가지는 방법이다. 주소 차용 요청에 대해 하나 혹은 그 이상의 예약 주소를 가진 노드들은 차용 요청에 대해 자신의 주소 중 하나를 주고 확인 메시지를 받은 후 자신의 주소 슬롯에서 그 예약 주소를 제거한다. 그리고 다른 노드의 주소 차용 요청 메시지의 응답을 전달하는 노드들은 차용 요청에 대한 첫 번째 응답에 대해서만 전달하고 다른 응답들은 자신의 예약 주소로 취한다. 이런 과정 후에도 도착된 응답들의 주소는 모두 차용을 요청한 에이전트 노드가 흡수한다. 만일 예약 슬롯의 다 채우고 남는 주소가 생긴다면, 해당 주소를 원래 노드에게 되돌려 준다. 그런 다음 에이전트 노드는 흡수한 주소들을 이용해서 주소를 할당할 수도 있고, 예약 주소를 줄 수도 있다. 예약 주소가 충분한 에이전트 노드는 DAD를 할 필요가 없는 경우도 생기게 된다. 주소 설정 과정이 끝난 후 에이전트 노드는 더 이상 예약 주소가 없을 때만 낙천적 DAD를 수행한다. 즉, 다중 예약 주소 슬롯은 전체적인 DAD 숫자를 줄이는 데 도움이 된다.

3.4 새로운 노드의 고장 혹은 이동

주소 설정 과정에서 제안된 기법은 비정상적인 경우들을 겪게 될 수도 있다. 제안된 프로토콜은 주소를 할당하기 위해 여러 메시지들을 사용하는데, 이런 메시지들이 주소 설정 과정 중 소실이 될 경우가 있다. 이럴 경우, 제안된 기법 또한 다른 프로토콜들의 대책과 마찬가지로 타이머를 이용해 메시지 분실을 해결한다.

만일 새로운 노드 i가 에이전트 노드를 찾는 요청을 한 후에 네트워크를 떠나거나 고장이 난다면, 에이전트 노드 j는 응답을 그냥 보낼 것이다. 주소 할당 요청을 다시 받기 전까지 노드 j는 노드 i가 에이전트를 찾는 과정에 대해서 관여하지 않는다. 그러므로 이러한 경우

아무런 문제가 되지 않는다. 그러나 노드 i가 주소를 구하는 요청을 한 후 없어지거나 고장이 발생한다면, 노드 j는 할당 주소가 올바르게 배달될 것인가를 확인하기 위해 타이머를 작동시키고 그에 대한 승인 메시지를 기다린다. 만일 타이머가 종료되기 전에 승인 메시지를 받지 못한다면, 노드 j는 이러한 과정을 몇 번 더 시도해야 한다. 그러한 시도가 모두 실패하면, 할당을 포기한다. 또 다른 문제는 노드 i가 노드 j로부터 자신의 예약 주소를 받지 않고 이동해 버리는 경우인데, 이럴 때 노드 j는 마찬가지로 여러 번 재시도를 한다. 이 또한 궁극적으로 실패한다면, 노드 j는 예약 주소를 주는 것을 포기하고, 노드 i는 2번 형태의 모바일 애드 혹 네트워크 노드가 된다.

3.5 에이전트 노드의 고장 혹은 이동

노드 j가 새로운 노드 i의 에이전트 노드가 되겠다는 응답을 보낸 후 혹은 주소 할당 요청을 받은 후에 노드 j가 네트워크에서 없어지거나 고장이 발생했다고 가정하자. 노드 i는 노드 j로부터 응답을 받기 위해 일정 시간 기다려야 한다. 만일 모든 시도가 실패하면, 노드 i는 새로운 에이전트를 찾는다. 이외에도 노드 j가 예약 주소를 주지 않고 사라진 경우에도 노드 i는 일정 시간 기다려야 한다. 노드 i가 예약 주소를 얻는 데 실패하게 되면, 노드 i는 2번 형태의 모바일 애드 혹 네트워크 노드가 된다.

3.6 같은 주소에 대한 사용 허가 요청

에이전트 노드들이 동시에 같은 주소를 얻기를 원하는 경우를 고려해보자. 즉, 두 개 이상의 에이전트가 우연히 똑 같은 주소를 선택했다면, 우리는 한 노드에게만 해당 주소를 사용할 수 있게 하고 다른 노드들에게는 부정적 메시지를 보내야 한다. 경쟁 노드들 중에서 특정한 노드에게 그 주소 사용을 허가하기 위해 가장 우선적으로 DAD의 시도 회수를 확인해봐야 한다. 일반적으로 DAD는 하나의 주소를 얻기 위해 3번 반복 시도한다. 따라서 어떤 노드에게 주소를 줄 것인가에서 DAD의 숫자가 높은 노드를 선택하는 것이 좋을 수 있다. 만일 이 숫자가 같다면, 어떤 에이전트 노드가 할당에 많이 참여 했는가를 나타내는 *allocation_counter*를 보고 결정한다. 이 변수는 할당에 참여할 때마다 증가시킨다. 만일 이 변수고 동일하다면, 낮은 주소를 가지고 있는 노드에게 우선권을 준다[9].

4. 네트워크 분리와 통합

모바일 애드 혹 네트워크 노드들은 임의로 움직이기 때문에, 네트워크 토폴로지가 쉽게 변하고 또한 네트워크 분리와 병합 문제가 발생한다. 주소 설정 문제에서는 주소 충돌을 일으키지 않는 네트워크 분리는 비교적 간단한 문제이다. 그러나 이에 비해 두 개 이상의 네트워크가 병합될 때, 주소 충돌이 발생할 수 있다. 왜냐하면 각각의 네트워크에서 할당된 주소들의 유일성은 해당 네트워크 안에서만 보장된 것이기 때문이다. 대부분의 네트워크 병합과 분리를 다루는 프로토콜들은 네트워크 분리와 병합을 발견하기 위해 그룹 ID를 사용한다. 네트워크 분리와 병합을 발견하기 위해, 주기적으로 하나의 리더 노드가 자신의 존재를 알리기 위해 전체 네트워크에 그룹 ID를 포함한 메시지를 브로드캐스트 한다. 만일 서로 다른 그룹 ID를 가진 두 개 이상의 노드가 만나게 된다면, 네트워크가 병합 되었다는 것을 판단할 수 있다. 또한, 하나의 노드가 더 이상 리더 노드로부터 그룹 ID가 포함된 메시지를 받지 못하면 네트워크 분리를 발견할 수 있다[9,10]. 네트워크 분리가 발생하면 분리된 네트워크에서는 새로운 리더 노드를 선출해야 한다.

네트워크 분리와 병합을 탐지하기 위해, 본 논문에서도 기존의 방법들과 같이 그룹 ID를 사용하는 방법을 선택한다. 본 논문에서는 병합을 탐지한 후 충돌한 주소를 찾고 이를 해결하는 방안을 다음과 같이 제안한다.

단순 브로드캐스트 방법: 첫째, 중복된 주소들을 찾기 위해 간단하게 각각의 노드가 자신의 주소와 자신의 예약 주소를 포함한 메시지를 브로드캐스트 하는 것이다. 그 메시지를 받은 모든 노드들은 자신의 주소와 충돌이 있는가를 확인한다. 만일 중복된 주소들이 있다면, 낮은 우선 순위를 가진 노드들은 그들의 주소를 변경해야 한다(우선 순위는 *allocation counter*를 이용하고 이것이 같을 때는 주소가 낮은 것에게 우선권을 준다). 이때, 자신의 실제 주소와 예약 주소가 모두 중복된 경우에는 해당 노드는 새로 네트워크에 들어온 노드와 같이 행동한다. 만일 예약 주소가 중복된 경우에는 예약 주소를 포기하고 2번 형태의 노드가 된다. 만일 사용하고 있는 주소만 중복이 된 경우, 해당 노드는 자신의 예약 주소를 이용해서 스스로 할당 받고 2번 형태의 노드가 된다. 이러한 방법은 매우 간단하지만, 모든 노드가 이와 같은 과정을 반복해야 하기 때문에 너무 오버헤드가 발생한다.

주소 수집가 방법: 둘째, 네트워크의 모든 주소들을 모으는 수집가(collector) 개념을 생각해 볼 수 있다. 서로 다른 네트워크들이 인접하게 되어 병합이 발생하면, 병합을 발견한 최초의 노드들이 생기게 되는데 이를 주소 수집가(address collector)라고 부르고 이는 각각 이전의 모바일 애드 혹 네트워크의 주소들을 수집하는 역할을 한다. 수집가 노드는 하나의 메시지를 브로드캐스트 함으로써 주소를 수집할 수 있다. 모든 노드는 이에 대해 자신의 주소와 예약 주소들의 포함해서 응답해야

한다. 마지막으로 수집가는 모든 노드들로부터 받은 응답을 가지고 주소와 예약 주소를 포함하는 테이블을 생성할 수 있다. 수집가 노드가 주소를 수집하기 위해 메시지를 브로드캐스트 하는 방법 외에 깊이 우선 탐색(DFS: Depth First Search) 알고리즘을 이용하는 방법을 제안한다. 모든 주소가 담긴 엔트리를 만들기 위해서는 모바일 애드 혹 네트워크에 있는 모든 노드들을 방문해야 한다. 따라서 수집가는 자신의 주소와 예약 주소를 포함한 엔트리가 담긴 메시지를 자신의 이웃 노드들에게 전달한다. 이 메시지를 받은 이웃 노드들은 자신의 주소와 예약 주소를 엔트리에 추가하고 엔트리에 존재하지 않는 이웃 노드들에게만 메시지를 전달한다. 메시지를 받은 노드는 이전 노드에게 반드시 승인 메시지로 응답해야 한다. 만일 모든 이웃들이 엔트리에 들어가 있다면, 노드는 자신의 이전 노드에게 승인 메시지 대신에 엔트리를 포함한 메시지를 다시 돌려보낸다. 이것을 백트래킹이라고 부른다. 궁극적으로 수집가 노드는 모든 주소가 담긴 엔트리를 결과로 받게 될 것이다. 이밖에, 또 다른 수집 방법은 한 홉 이내의 이웃에게만 브로드캐스트 하는 방법이다. 수집가는 주소 엔트리를 만들기 위해 자신의 한 홉 이내의 이웃 노드에게만 메시지를 보낸다. 하나의 노드가 이 메시지를 받았을 때, 노드들은 자신의 주소를 추가하고 그 엔트리를 수집가에게 보고 하거나 자신의 한 홉 이내의 이웃 노들에게만 다시 브로드캐스트 한다. 만일 자신의 이웃 노드들이 이미 모두 엔트리에 들어가 있다면, 수집가 노드에게 현재까지의 노드들이 담긴 엔트리를 보고 한다. 그러나 아직 포함되지 않은 노드가 있다면 그 노드들을 엔트리에 포함해 다시 브로드캐스트 해야 한다. 이를 통해 수집가 노드는 모든 주소를 얻을 수 있다. 수집가 노드들은 모든 주소를 모은 후에, 충돌 주소를 찾기 위해 각각의 리스트를 비교한다. 만일 충돌 주소가 발견되면, 해당 노드에게 주소 충돌을 알려줘야 한다. 우선 순위가 낮은 그룹의 노드에게만 주소 충돌을 알려줘 주소를 변경하게 한다. 원래 주소가 충돌인 경우, 예약 주소가 충돌인 경우, 둘 다 충돌인 경우가 발생할 수 있는데, 이는 단순 브로드캐스트 방법과 같은 방법으로 해결하면 된다.

5. 성능 분석

본 논문에서는 주소 자동 설정의 중요한 3가지 요구사항인 할당된 주소의 유일성, 통신 오버헤드, 주소 할당 시간의 관점에서 모의 실험을 수행하였다. 제안된 방법은 예약 주소 슬롯이 4개 일 경우 가장 좋은 성능을 낸다는 것을 확인하였다. 제안된 방법은 Strong DAD, MANETconf, Prophet과 비교하였다.

가장 적절한 예약 주소 슬롯을 찾는 것과 할당된 주소의 유일성의 결과를 보이기 위해서 자바 프로그램을 이용하였다. 왜냐하면, ns-2를 가지고 실험하는 경우 노드 수가 100개를 넘어가게 되면 모의 실험에 많은 어려움을 겪기 때문이다. 주소 할당 시간과 통신 오버헤드에 대해서는 ns-2를 이용하였다.

5.1 모의 실험 환경

Java 모의 실험 환경: 자바 프로그램을 이용한 모의 실험은 다음과 같이 정형화할 수 있다. Prophet을 위한 네트워크 내의 모든 노드들 중에 하나의 에이전트를 임의로 선택하고, 제안된 방법을 위해서는 선택된 이웃 노드들 중에 임의의 에이전트 노드를 선택한다. 이웃 노드의 수는 1과 8사이의 균일 분포를 따른다. 노드의 이동성은 고려되지 않았으나, 노드의 임의 선택이 이동성의 효과를 대신할 수 있을 것으로 본다. 이전 노드를 위한 주소 할당이 완료된 후, 새로운 노드가 네트워크에 도착하는 것으로 가정한다. 모든 노드는 네트워크를 떠나지 않기 때문에 노드 수는 계속 증가한다.

Ns-2 모의 실험 환경: 본 논문에서는 수정된 랜덤 웨이 포인트 모델이 반영된 버전 2.27인 ns-2를 사용하였다[20-22]. 노드의 최대 속도는 5 m/s, 최소 속도는 1m/s, 멈춤 시간은 10 초이다. 표 1은 2가지의 모의 실험 환경을 보여준다. 새로운 노드들 간의 도착 시간 간격은 [0, 10]의 균일 분포를 따른다. 잘 연결된 네트워크를 만들기 위해, 모의 실험을 시작하기 전에 미리 설정된 노드들을 네트워크에 추가하였다. 네트워크 병합과 분리는 모의 실험에서 고려되지 않았고, 모의 실험 동안에 제안된 방법의 예약 주소 슬롯 수는 4로 고정하였다.

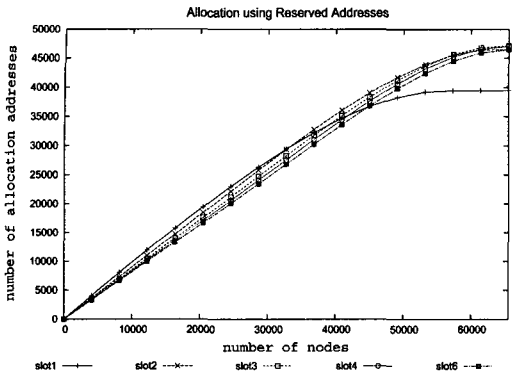
표 1 모의 실험 파라미터들

Parameters	Environment I	Environment II
The number of nodes	35, 40, ···, 60	50, 60, ···, 100
Preconfigured nodes	30	40
Area	750m x 750m	1000m x 1000m
Simulation time	900 seconds	900 seconds
Address range	1~65534 (16 bits)	1~254 (8 bits)
Routing protocol	AODV	AODV

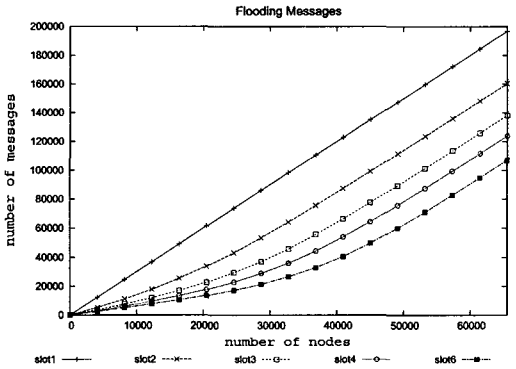
5.2 최적의 예약 주소 슬롯 수

새로운 주소는 예약 주소, 차용 주소 혹은 DAD를 통해 할당될 수 있다. 제안된 방법은 하나의 노드가 예약 주소를 통해 주소를 할당 받을 때 가장 좋은 성능을 보이는 것은 매우 명백하다. 그러므로 가장 적합한 예약 주소 슬롯 수를 찾기 위해서 예약 슬롯 수를 1, 2, 3, 4 그리고 6으로 변화시켜 가며 모의 실험을 하였다. 이때 주소의 범위는 65536개로 하였다.

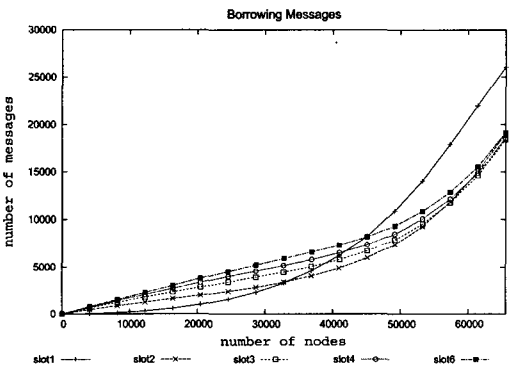
그림 5(a)는 예약 주소를 이용한 할당 수를 보여준다. 노드 수가 33,000개보다 작은 경우면, 한 개의 슬롯을



(a) 예약 주소들에 의해 할당된 주소 수



(b) 플러딩 메시지 수



(c) 차용 메시지 수

그림 5 예약 슬롯과 다른 요소들과의 관계

가진 경우가 가장 좋은 결과를 보인다. 그러나, 그 지점을 지나면, 2번 형태의 노드 수 증가로 인해 한 개의 슬롯을 가진 제안된 방법은 예약 주소로부터 할당 받는 수가 줄어든다. 슬롯 수가 2, 3, 4, 혹은 6일 때는, 그 결과가 매우 유사하다. 슬롯 수가 늘어남에 따라, 예약 주소로부터의 할당 수는 상대적으로 점점 작아진다. 왜냐하면, 슬롯이 커지면 각 노드가 보유하는 예약 주소들의 표준 편차도 커지기 때문이다.

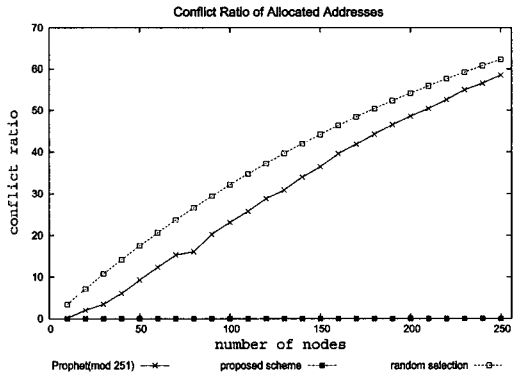
그림 5(b)와 (c)는 슬롯 수와 플러딩 메시지 사이의 관계, 그리고 차용 메시지와 슬롯 수의 관계를 각각 보인다. DAD의 수는 슬롯 수에 반비례하기 때문에, 슬롯 수가 증가함에 따라 플러딩 메시지는 감소한다. 반대로, 차용 메시지 수는 증가한다. 메시지 수와 할당 시간을 고려하여 슬롯 수로 4가 가장 적합하다고 판단했다.

5.3 할당된 주소들의 유일성

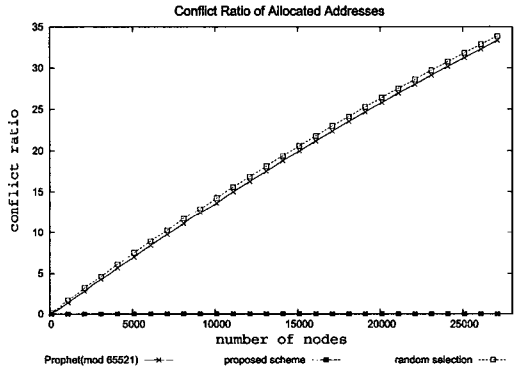
주소 충돌은 라우팅 프로토콜과 응용에 비정상적 행동을 야기할 수 있기 때문에, 할당된 주소의 유일성을 보장하는 것은 가장 중요한 요구 사항 이다. 우리는 자바 프로그램을 사용해서 제안된 방법, Prophet, 그리고 임의 할당 방법 각각의 충돌율을 256과 65536의 주소 범위 상에서 비교하였다. 이 때, 충돌율은 중복 주소를 가진 노드 수를 전체 노드 수로 나누어 구했다. 임의 할당 방법에서, 새로운 노드는 주소 범위보다 작은 균일 분포를 가지는 임의 수를 선택하여 자신의 주소로 취한다. Prophet에서 각각의 노드는 양의 정수를 생산하는 유일한 모듈러 함수를 가진다. 256 주소 범위에 대해서, 주소 = $a + \prod_{i=1}^{k} p_i^{e_i} \bmod 251 + 1$, a는 첫 번째 노드에 대한 임의의 주소이다. 소수 p_i는 $p_1 < p_2 < \cdots < p_k$를 만족하고, 만일 k가 209이면, p_k는 1291이며, 209번째 소수를 의미한다. 주소 범위 65536일 때는, 주소 = $a + \prod_{i=1}^{k} p_i^{e_i} \bmod 65521 + 1$ 이다.

즉 각 노드의 함수는 address = $a + cp_i \bmod 251 + 1$로 표현될 수 있다. 여기서 c는 상수이다. 이것은 선형 콩그루엔셜 재생기의 형태를 띤다[23]. 만일 우리가 2^m과 같은 소수가 아닌 수 N을 제수로 사용한다면, 충돌율은 증가한다. 왜냐하면, 만일 수열을 위해 사용되는 피제수가 N으로 나누어 진다면, 그 수열의 주기가 줄어들기 때문이다. 사용된 제수 251과 65521은 주소 범위에서 가장 가까운 소수이다.

Prophet은 주소 충돌을 확인하는 방법을 가지고 있지 않다. 유일성은 주소 할당 과정에서 반드시 보장되어야 한다. 주소 범위가 한정적이기 때문에, 여러 순열들 사이에서 두 개 혹은 그 이상의 같은 수가 할당 될 수 있다. 그림 6은 노드 수에 따른 충돌율을 보여준다. 주소 범위가 256일 때, Prophet은 네트워크의 전체 노드 수가 작은 경우에도 매우 많은 충돌을 보인다. 주소 범위가 65536일 때, 충돌 수는 줄어들지만, 여전히 Prophet을 주소 설정 방법으로 사용하기에는 충돌 수가 여전히 많다. 주소 비트가 증가함에 따라, 충돌율은 임의의 주소 할당 방법에 근접한다. 결과적으로, Prophet은 매우 큰 주소 범위를 필요로 하고 전체 네트워크의 노드 수

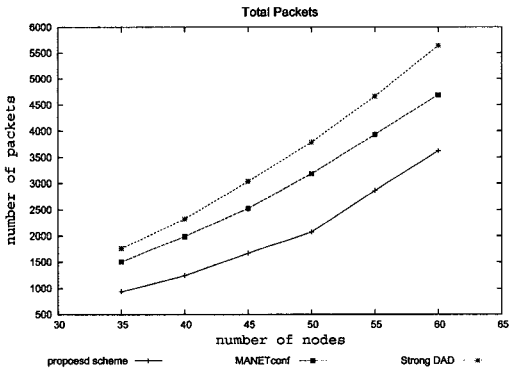


(a) 256 Address Range

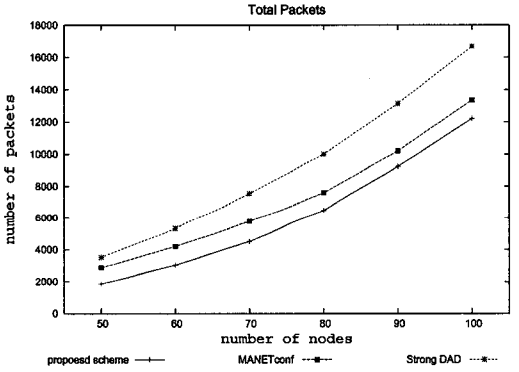


(b) 65536 Address Range

그림 6 할당된 주소들의 충돌율



(a) Environment I



(b) Environment II

그림 7 통신 오버헤드

는 작아야 한다. 이것은 주소 공간의 낭비이다. 이와 달리 제안된 방법은 DAD에 의해 할당된 주소들 중에 충돌을 허락하지 않는다.

5.4 통신 오버헤드

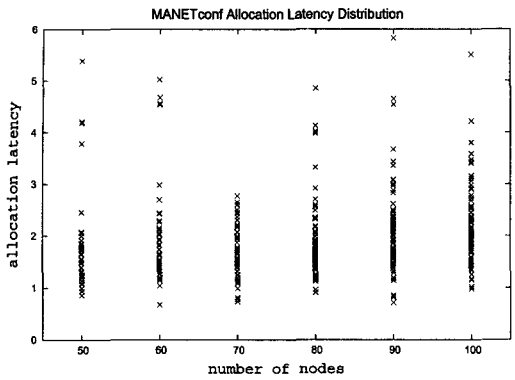
우리는 ns-2를 이용해서 제안된 방법과 Strong DAD, MANETconf를 비교하였다. 그림 7은 노드 수에 따른 패킷 전송 수를 보여준다. 그 결과는 DAD를 고용한 세가지 방법들은 패킷의 수가 노드 수에 비례한다는 것을 보여준다.

DAD 과정 동안, Strong DAD는 세 번의 플러딩을 하고, MANETconf는 한번의 플러딩을 한 후 응답하지 않은 노드들에게 유니캐스트를 통해 다시 응답을 받는다. 제안된 방법은 3번의 플러딩을 한다. Strong DAD와 MANETconf는 이런 DAD를 할당 되지 않은 주소를 얻을 때까지 계속한다. 반면에, 제안된 방법은 낙천적 DAD를 행한다. 이는 주소를 얻는 데 성공하느냐 마느냐에 관계없이 오직 한번만 수행한다. Strong DAD와 MANETconf는 새로운 노드가 네트워크에 들어올 때 한번 이상의 DAD를 하지만, 제안된 방법은 새로운 노드에게 주소를 할당 한 후 노드들을 위한 예약 주소가 없을 때만 DAD를 한번만 수행한다. 그러므로 Strong DAD나 MANETconf 보다 DAD 수가 줄어들 것이다. 그림 7에서 Strong DAD와 제안된 방법의 차이는 노드 수가 증가함에 따라 줄어든다. 왜냐하면, 예약 주소를 통해 바로 할당되는 경우보다 플러딩이 요구되는 주소 차용에 의해 할당되는 주소 수가 노드 수에 비례해서 증가하기 때문이다.

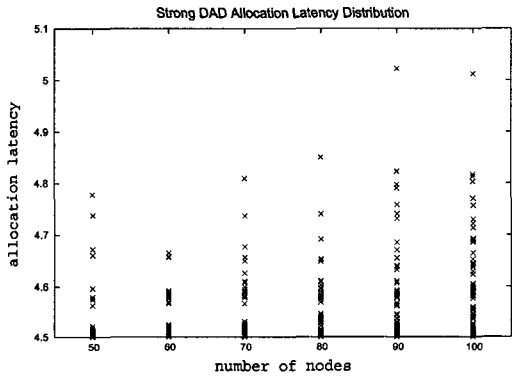
5.5 할당 시간

주소 할당 시간을 측정하기 위해 ns-2를 이용해서 모의 실험을 수행하였다. 그림 8은 각각의 방법의 주소 할당 시간의 분포를 보여준다. 각각 (a) MANETconf, (b) Strong DAD, (c) 제안된 방법을 나타낸다. 이 모의 실험은 환경 I 상에서 수행된 것이다. 그림 8(a)는 2초와 3초 사이의 할당 시간을 가진 노드들이 가장 많고, (b)는 대부분의 노드들이 4.5초 근처에서 주소를 할당받은 것을 볼 수 있다. 마지막으로 제안된 방법은 2초이내에서 주소를 얻는 데 성공한 것을 볼 수 있다.

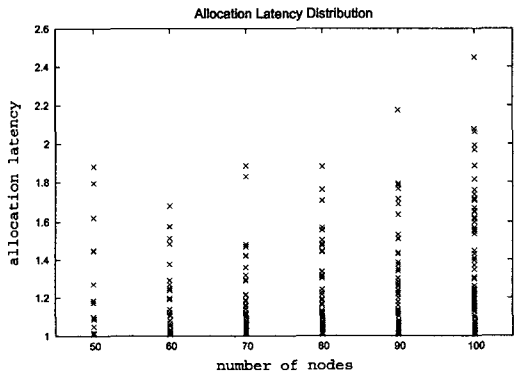
그림 9는 실험 환경 I과 II상에서의 각각의 평균 주소



(a) MANETconf



(b) Strong DAD

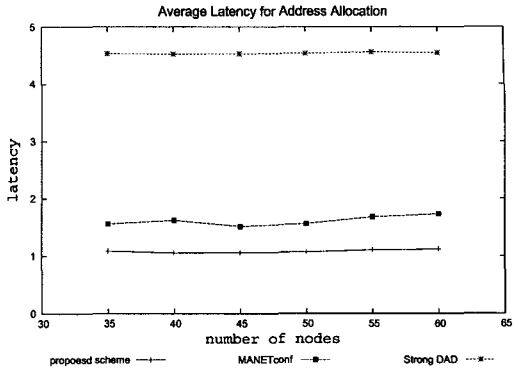


(c) Our Scheme

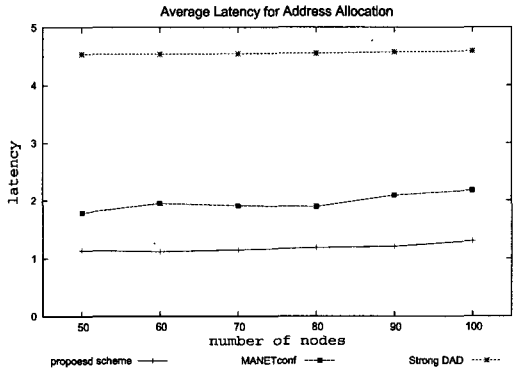
그림 8 할당 시간 분포

할당 시간을 비교한 것을 보여준다. MANETconf와 제안된 방법에서 에이전트를 찾는데 걸리는 시간은 각각 1초로 설정하였고, DAD를 위한 타임 아웃은 1.5초로 설정했다.

Strong DAD는 3번의 DAD 과정이 다 끝날 때까지 기다리기 때문에, 주소를 얻는데 최소 4.5초가 걸린다.



(a) Environment I



(b) Environment II

그림 9 평균 할당 시간

MANETconf는 첫 번째 DAD 과정에서 응답하지 않은 노드들에게만 다시 요청하기 때문에, Strong DAD 보다 빠른 시간에 주소를 얻는다. 제안된 방법은 모든 구간에서 다른 비교 대상들 보다 뛰어남을 볼 수 있다. 예약 주소와 주소 차용 기법이 이를 가능하게 한 것이다. 제안된 기법은 새로운 노드가 자신의 에이전트를 찾는데 1초의 시간을 기다리며, 그리고 하나 이상의 예약 주소를 가진 에이전트를 선택한다. 따라서, 평균 할당 시간은 1초 근방이 된다. 노드 수가 증가함에 따라, 제안된 방법의 평균 할당 시간은 약간 상승한다. 모의 실험 환경 I에서는 환경 II보다 주소 범위가 작고 노드 상대적으로 노드 수도 많으므로, 모든 방법들이 상대적으로 할당 시간이 조금 길어 진다. 특히 MANETconf의 경우는 I과 II에 따라 상대적으로 다른 기법들보다 차이를 더 많이 보인다.

6. 결론 및 향후 연구 방향

모바일 애드 혹 네트워크에서 주소를 설정하기 위해, 본 논문에서는 주소 자동 설정의 중요한 요구 사항들을

- 짧은 주소 할당 시간, 낮은 통신 오버헤드, 할당된 주소들의 유일성 보장 - 만족시키기 위해 예약 주소와 낙천적 DAD를 이용한 분산형 자동 주소 설정 방법을 제안하였다. 예약 주소는 주소 할당 시간을 줄이는 데 도움을 주었고, 주소 차용 방법은 새로운 노드가 에이전트 노드로부터 직접 주소를 얻지 못하는 경우에도 상대적으로 빠른 주소 할당을 보장하도록 해 주었다. 게다가, 두 개 이상의 예약 주소를 갖는 방법과 낙천적 DAD은 통신 오버헤드를 줄이고 할당된 주소의 유일성을 보장하였다. 제안된 방법은 예약 주소 슬롯이 4개 일 때, 네트워크 트래픽과 주소 할당 시간에 있어서 좋은 성능을 보이는 것을 모의 실험을 통해 보였다. 또한, 주소 수집가 개념을 도입하여 네트워크 분리와 통합에 관한 해결책을 제시하였다.

향후에, 본 논문의 모의 실험에서 다루지 못한 네트워크 분리와 통합에 관한 환경을 고려해 모의 실험을 추가할 것이다. 또한, 제안된 방법을 보다 안전하게 만들기 위해, 모바일 애드 혹 네트워크에서의 보안 분야에 관한 연구를 수행할 것이다. 네트워크에 대한 공격이 점점 지능화되어 가고 또한 그 피해가 매우 심각해져 가므로 주소 설정 방법들도 보안에 대한 구체적인 해결책이 필요하다.

참 고 문 헌

[1] V. D. Park and M. Scott Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," in IEEE Conference on Computer Communications (Infocom '97), 1997.

[2] Charles Perkins and Elizabeth Royer, "Ad Hoc On-Demand Distance Vector Routing," in 2nd IEEE Workshop on Selected Areas in Communication, February 1999, pp. 90 - 100.

[3] M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1395-1414, August 1999.

[4] R. Droms, "Dynamic Host Configuration Protocol," Network Working Group-RFC 2131, March 1997.

[5] Mesut Günes and Jörg Reibel, "An IP address configuration Algorithm for Zeroconf. Mobile Multi-hop Ad-hoc Networks," In Proceedings of the International Workshop on Broadband Wireless Ad-Hoc Networks and Services, Sophia Antipolis, France, September 2002.

[6] Toner,S. & O'Mahony,D., "Self-Organizing Node Address Management in Ad-hoc Networks," in Springer Verlag Lecture notes in Computer Science 2775, Springer Verlag, Berlin, 2003, pp 476-483

[7] Yuan Sun and Elizabeth M. Belding-Royer, "Dynamic Address Configuration in Mobile Ad hoc Networks," UCSB Technical Report 2003-11, June 2003.

[8] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Elizabeth M. Belding-Royer and Yuan Sun, "IP Address Autoconfiguration for Ad Hoc Networks," draft-ietf-manet-autoconf-01.txt, November 2001.

[9] Sanket Nesargi, Ravi Prakash, "MANETconf Configuration of Hosts in a Mobile Ad Hoc Network," Proc. of IEEE Infocom, June 2002.

[10] Hongbo Zhou, Lionel Ni, Matt Mutka, "Prophet Address Allocation for Large Scale MANETs," In Proceedings of the 22th IEEE Conference on Computer Communications (INFOCOM'03), March 2003.

[11] A. Misra, S. Das, A. McAuley, and S. K. Das, "Autoconfiguration, Registration and Mobility Management for Pervasive Computing," IEEE Personal Communications (Special Issue on Pervasive Computing), Volume 8, Issue 4, pp. 24-31, Aug 2001.

[12] Mansoor Mohsin and Ravi Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," IEEE Military Communications Conference (MILCOM 2002), October 2002.

[13] T. Mansi Ramakrishnan, "A Protocol for Dynamic Configuration Of Nodes in MANETs," Master's thesis, Computer Science, University of Texas at Dallas, August 2002. (advisor: Ravi Prakash)

[14] K. Weniger, M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks," Proceedings of European Wireless 2002, Florence, Italy, Feb. 2002.

[15] K. Manousakis, A. McAuley, R. Morera, J. Baras, "Routing Domain Configuration for More Efficient and Rapidly Deployable Mobile Networks," Army Science Conference, Dec. 2002.

[16] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad hoc Networks," In Proceedings of IEEE WCNC 2003, New Orleans, USA, Mar. 2003.

[17] Nitin Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), June 2002.

[18] P. Patchipulusu, "Dynamic Address Allocation Protocols for Mobile Ad Hoc Networks," Master's thesis, Computer Science, Texas A&M University, August 2001. (advisor: Nitin H. Vaidya)

[19] Jeff Boleng, "Efficient Network Layer Addressing for Mobile Ad Hoc Networks," In Proceedings of 2002 International Conference on Wireless Networks (ICWN'02), pp. 271-277, June 2002.

[20] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-

Hop Wireless Ad Hoc Routing Protocols," Proceedings of the Fourth Annual ACM/IEEE Inter-national Conference on Mobile Computing and Networking, pp. 85 - 97, October 1998.
[21] J. Yoon, M. Liu and B. Noble, "Sound Mobility Models," in Proc. ACM MobiCom, September 2003, San Diego, CA.
[22] J. Yoon, M. Liu and B. Noble, "Random Waypoint Considered Harmful," in Proc. IEEE INFOCOM, vol 2, pp 1312-1321, April 2003, San Francisco, CA.
[23] Raj Jain, "The Art of Computer Systems Performance Analysis," John Wiley & Sons, New York, pp 439-440, 1991.

김 남 훈

1996년 2월 숭실대학교 전자계산학과 공학사. 1998년 2월 숭실대학교 전자계산학과 공학석사. 2000년 3월~2003년 1월 팬타 시큐리티 시스템㈜ IDS 개발팀 주임연구원, 개발팀장. 1998년 3월~현재 한국정보통신대학교 공학부 박사과정

안 소 연

1997년 2월 한국과학기술원 공학사. 1997년 2월~1997년 10월 시스템공학연구소 연구조원. 1997년 11월~1998년 2월 ㈜아이시티 프로그래머. 2000년 2월 한국정보통신대학교 공학석사. 2000년 3월~현재 한국정보통신대학교 공학부 박사과정

문 경 덕

1990년 2월 한양대학교 전산학 학사. 1992년 2월 한양대학교 전산과 석사. 1992년 3월~현재 한국전자통신연구원 선임연구원. 2001년 1월~현재 한국전자통신연구원 홈네트워크 미들웨어연구팀장

이 영 희

1976년 2월 서울대학교 공과대학 공업교육학과 공학사. 1980년 2월 서울대학교 공과대학 공업교육학과 공학석사. 1984년 6월 프랑스 UTC 전산학 박사. 1984년 8월~1997년 12월 ETRI, 정보통신표준연구센터 센터장. 1986년 7월~1987년 11월 IBM T.J.Watson 연구소 초빙과학자. 1998년 1월~현재 ICU 교수(공학부장, SOTI 연구소장 역임)