이동 애드 혹 네트워크의 Mobile IP 지원을 위한 네트워크 구조 및 라우팅 프로토콜

정회원 오 훈*, 준회원 판 안 탄**

Network Architecture and Routing Protocol for Supporting Mobile IP in Mobile Ad Hoc Networks

Hoon Oh* ***Regular Member***, Anh TanPhan** ***Associate Member***

요 약

이동 애드혹 네트워크에 Mobile IP를 효율적으로 지원하기 위하여 인프라구조 네트워크와 이동 애드혹 네트워크의 트리 기반 통합 네트워크 구조를 제안하고, 통합 네트워크의 형성 및 관리를 위한 네트워크 구조 관리 프로토콜 및 통합 네트워크에 특화된 트리 기반 라우팅 프로토콜을 제안한다. 통합 네트워크는 두 이기종 네트워크를 연결하는 고정 인터넷 게이트웨이 (IG)를 가지고 있으며, 이동 노드들은 IG와 직접 통신 거리에 있는 이동 노드들을 중심으로 작은 트리들을 형성한다. 새로운 노드는 먼저 임의의 트리에 가입하고 트리를 따라 자신의 홈 에이전트 (HA) 및 외부 에이전트 (FA)에 등록한다. 또한, 제안하는 라우팅 프로토콜은 트리 정보를 이용하여 효율적으로 경로를 설정한다. 주요 네트워크 전개 시나리오에 대한 시뮬레이션을 통해서 트리 기반 통합 네트워크의 효율성을 분석하였으며, 제안하는 라우팅 프로토콜을 기존의 Mobile IP 지원 AODV 프로토콜과 성능을 비교 분석하였다.

Key Words : MANET, Mobile IP, Tree Structure, Mobility Management, Routing

ABSTRACT

We propose a tree-based integrated network of infrastructure network and mobile ad hoc network to effectively support Mobile IP for mobile ad hoc networks and also proposed a network management protocol for formation and management of the integrated network and a tree-based routing protocol suitable for the integrated network. The integrated network has fixed gateways (IGs) that connect two hybrid networks and the mobile nodes in the network form a small sized trees based on the mobile nodes that are in the communication distance with a IG. A new node joins an arbitrary tree and is registered with its HA and FA along tree path. In addition, the proposed protocol establishes a route efficiently by using the tree information managed in every node. We examined the effectiveness of the tree-based integrated network for some possible network deployment scenarios and compared our routing protocol against the Mobile IP supported AODV protocol.

Ⅰ. 서 론

이동 애드 혹 네트워크 (MANET: Mobile Ad Hoc Networks)는 고정 인프라구조가 없이 복수의 이동 단말기(노드) 들이 임시로 구성하는 무선 통신 네트워크이다. 그 유용성에도 불구하고 이동 노드들이

※ 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2005- 003-D00222)
※ 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업으로부터 부분적인 지원을 받았음 (IITA-2007_C1090-0701-0039)
* 울산대학교 UbiCom 연구실 (hoonoh@ulsan.ac.kr), ** 울산대학교 UbiCom 연구실
논문번호 : KICS2007-08-382, 접수일자 : 2007년 8월 5일, 최종논문접수일자 : 2007년 12월 18일

인터넷이나 다른 이동 애드 혹 네트워크에 속하는 노드들과 통신을 원하거나 혹은 그들로부터 연결 요청을 받는 것과 같은 외부 네트워크와의 연결성이 요구되는 분야에 그 응용성이 크게 제한된다.

최근 몇 년간 통합 네트워크를 위한 이동성 관리 및 라우팅 프로토콜에 대한 많은 연구가 있었다. Broch et al은 인터넷 게이트웨이 (IG)가 두 개의 인터페이스를 갖는다고 가정하였다[1]. 표준 IP 라우팅 메커니즘을 실행하는 하나는 인터넷 노드들과의 통신을 담당하며, 이동 애드 혹 네트워크 라우팅 메커니즘을 실행하는 다른 하나는 이동 노드들과 통신한다. 이동 애드 혹 네트워크 라우팅을 위하여 DSR[2]를 사용하였다. Jonsson et al 은 이동 노드들이 MIPMANET 셀 스위칭 알고리즘을 사용하여 외부 에이전트 (FA)에 언제 등록할 것인지를 결정하며, 이동 애드 혹 네트워크 라우팅을 위하여 AODV[4]를 사용하였다[3]. Sun et al은 Mobile IP와 AODV가 이동 노드와 게이트웨이 사이에 복수 홉 경로를 찾기 위한 협력 방법을 제안하였다[5]. Prashant et al은 에이전트 광고, 엿듣기, TTL 설정, 에이전트 탐색 등과 같은 여러 기법들을 조합하는 하이브리드 방법을 제시하였다[6]. Ammari et al은 Mobile IP와 DSDV[7]를 사용하는 3계층 방식을 제안하였다[8]. 이동 애드 혹 네트워크의 인터넷 연결성을 제공하기 위하여 이동 게이트웨어를 사용하며 라우팅을 위하여 DSDV를 사용하였다. 이러한 방식들은 이동성 관리를 위하여 네트워크의 구조 관리보다는 이동성 관리 기법을 차별화하는데 초점을 두었으며, 라우팅을 위하여 기존의 이동 애드 혹 네트워크 라우팅 프로토콜을 수정하여 사용하였다. 따라서, 대부분 비효율적인 플러딩 기법을 사용하기 때문에 네트워크 콘트롤 오버헤드가 문제된다. 또한, 요구기반 라우팅 프로토콜의 경로 탐색을 위하여 여전히 비효율적인 플러딩에 의존하였다.

이 논문에서는 인프라구조 네트워크와 이동 애드 혹 네트워크의 효율적인 통합을 위하여 트리 기반의 통합 네트워크 구조를 제안하고, 통합 네트워크의 형성 및 관리 방안과 Mobile IP를 적용하기 위한 이동성 관리 방법을 논의하며, 제안된 트리 기반 통합 네트워크에 특화된 효율적인 트리 기반 라우팅 프로토콜을 제안한다. 트리 기반 통합 네트워크는 고정 인터넷 게이트웨이 (IG)와 직접 통신 거리에 있는 이동 노드를 루트 노드로 하여 형성되는 작은 트리들의 조합으로 이루어진다. 새로운 이동 노드는 트리에 가입하고 트리 경로를 따라 외부 에이전트 (FA) 및 홈 에이전트 (HA)에 가입하며, 트리에 가입된 이동 노드들은 자유롭게 트리를 떠날 수 있다. 이동 노드는 효율적인 인터넷 연결성을 위하여 임의의 IG까지 최단거리를 제공하는 트리에 가입한다. 트리에 가입한 노드는 노드의 이동성으로 인하여 임의의 IG까지 더 짧은 거리를 찾거나 혹은 링크가 파손되는 경우에 자신의 부모를 변경할 수 있다. 이런 식으로 토폴로지가 변경되면 변경을 감지한 노드는 정보 갱신 요청 (UREQ: Update Request) 메시지를 조상 노드들 혹은 후손 노드들에게 전송함으로써 토폴로지 정보를 갱신한다. 따라서, 트리의 크기를 가능한 작게 유지함으로써 네트워크 토폴로지 관리, 이동성관리, 경로설정에 관련된 오브헤드를 줄인다.

프로토콜의 설계 목표는 다음과 같다. 첫째, 경로 설정 지연시간을 줄인다. 트리 정보와 신뢰성이 높은 인프라구조 통신 자원을 최대한 이용함으로써 경로 설정에 걸리는 지연 시간을 줄일 수 있다. 두 번째, 이동 노드 간의 통신시에 인프라구조 네트워크의 자원 사용을 줄인다. 이동 노드의 수가 많아짐에 따라 인프라구조 네트워크가 과부하 되는 현상을 방지하고자 하는데 있다. 이 목표는 첫 번째 목표와 상충하지만, 통신 부하이나 통신의 효율성을 고려하여 인프라구조 네트워크와 이동 애드 혹 네트워크 사이에 적절하게 통신 오버헤드를 분배함으로써 두 가지 목표를 어느 정도 달성할 수 있다. 마지막으로, 통신 경로 설정에서 통신의 효율성을 최대한 고려한다. 예를 들면, 통신을 원하는 두 이동 노드가 너무 멀리 떨어져 있는 경우에는 무선 자원만을 사용하는 통신은 비효율적이다.

논문의 구성은 다음과 같다. 섹션 II는 배경, 섹션 III는 네트워크 구조 관리 프로토콜, 섹션 IV는 통합 네트워크에서의 라우팅 프로토콜을 설명한다. V장에서는 성능 평가를 하고 VI에서 결론을 맺는다.

Ⅱ. 배 경

2.1 네트워크 모델

고정 IG는 두 개의 네트워크 인터페이스 즉, 이동 애드 혹 네트워크와의 통신을 담당하는 IF1과 인터넷과의 통신을 담당하는 IF2를 갖는다. 두 개의 노드가 상호 통신 범위 내에 있을 때에는 “연결”되었다고 한다. 모든 이동 노드와 IG는 라우터 기능을 갖는다. 이동 노드들은 서로 다른 노드들 사이에 전송되는 패킷들을 엿들을 수 있다고 가정한다[9].

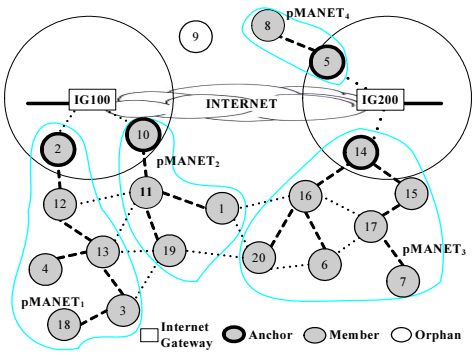


그림 4. 트리 기반 통합 네트워크 구조의 예

표 1. 트리 정보 구조 (Tree Information Table)

IG		
Anchor (닻)		
Parent (부모)		
Hop count to anchor (닻까지의 홉 수)		
Node status: 닻, 멤버, 고아		
Des 1	Des 1로 가는 다음 노드	Des 1까지 거리
Des 2	Des 2로 가는 다음 노드	Des 2까지 거리
…	…	…

* Des: descendant (트리 내의 후손 노드)

이 경우 네트워크는 비방향성 그래프 G로 모델된다. IG 및 이동 노드들은 "노드"로 모델되고, 두 노드 사이의 연결은 "간선(edge)"으로 모델된다. 모든 IG들의 집합을 V_1,이동 노드들의 집합을 V_2라고 하면 그래프 G = (V, E), V = $V_1 \cup V_2$ 그리고 E = $E_1 \cup E_2$이다. E_1 = {(x, y)|x∈V_1, y∈V_1, x와 y는 연결} 그리고 E_2 = {(x, y)|x∈V_2, y∈V_2, x와 y는 연결} ∪ {(x, y)|x∈V_1, y∈V_2, x와 y는 연결}.

2.2 용어 및 표기

▪ (**이웃 노드**) 노드 i의 이웃 노드의 집합, N_i = {x|(i, x)∈E}.

▪ (**pMANET**) **pMANET**는 V_2에 속한 루트 노드가 V_1에 속한 어떤 노드에 연결되어 있고 루트 노드에서 시작하여 V_2의 부분 집합으로 이루어지는 연결 트리이다. pMANET에 속한 노드는 *멤버* (Member), 멤버가 아닌 노드는 *고아* (Orphan), 그리고 멤버인 루트 노드는 *닻* (Anchor)이라고 명명한다.

▪ (**NeighborTable**) 모든 노드 i는 N_i에 속한 노드들에 대하여 **NeighborTable**$_i$를 관리한다. 이웃 노드 j의 엔트리는 NeighborTable$_i$(j) = (j의 ID, j의 관계, j의 IG, j의 닻, j에서 j의 닻까지 거리, j의 최종 갱신 시간)로 나타내며, "j의 관계"는 j가 i의 자식 노드인지 아닌지를 나타낸다.

▪ (**TreeInformationTable**) 모든 노드 i는 표 1에서와 같이 자신이 속한 트리의 정보를 관리할 수 있는 **TreeInformationTable**$_i$를 가진다.

▪ (**브로드케스트, 유니케스트**) **브로드케스트**는 어떤 노드가 이웃하는 모든 노드에게 메시지를 전달하는 것을 말하며 메시지 내에 특정 이웃 주소를 지정하지 않는다. 반대로 **유니케스트**는 특정 이웃 노드를 지정하여 전송하는 것을 의미한다.

이 논문에서는 어떤 노드가 특정 이웃 노드에 유니케스트하는 모든 메시지는 다른 모든 이웃 노드들이 엿들을 수 있다고 가정한다. 브로드케스트 메시지와는 달리 유니케스트의 경우에는 충돌 방지를 위한 MAC 계층의 CSMA/CA 전송 프로토콜이 적용된다. 본 논문에서는 트리의 정보를 이용하여 메시지를 전송할 때 대부분 유니케스트를 사용함으로써 충돌로 인한 채널 대역 폭의 낭비를 줄이고 전송 후에 MAC 계층의 ACK를 확인함으로써 전송 성공 혹은 전송 실패를 즉각적으로 확인한다.

그림 1의 통합 네트워크 예는 22개의 노드 (IG100, IG200, 1, 2, …, 20)로 구성되어 있다. 굵은 실선으로 된 간선과 점선으로 된 간선은 인프라 구조 네트워크와 이동 애드 혹 네트워크를 각각 나타낸다. 파선 (Dashed Line)으로 된 큰 원은 IG의 IF1의 무선 전송 범위를 나타내며 무선 이동 노드의 전송범위는 표시되어 있지 않지만 IF1의 그것과 동일하다. 빈 원, 그림자 원 그리고 굵은 그림자 원은 고아, 멤버 및 닻을 각각 나타낸다. 닻 2, 5, 10, 14는 네 개의 pMANET를 형성하며, 각 pMANET는 굵은 연결 파선으로 표시되어 있다.

2.3 메시지 정의

이 절에서는 이동성 관리 및 라우팅 프로토콜에서 사용할 메시지의 구조 및 사용 형태를 기술한다.

▪ **MN-Hello, IG-Hello**: 멤버 노드는 자신의 부모 노드에게 주기적으로 MN-Hello를 유니케스트한다 (고아 노드는 부모가 없으므로 전송하지 않는다). 루트 노드인 닻의 경우는 자신의 IG에게 주기적으로 전송한다. 다른 이웃 노드들은 엿듣기를 통해서 수신한다. MN-Hello = (MsgId, NodeId, IGId, AnchorId, ParentId, Over Capacity, HopsToAnchor)로 정의된다. MsgId는 메시지 고유 번호이다. <NodeId, MsgId> 조합에 의해서 특정 메시지는 유일하게 정의된다. IGId, AnchorId, 그리고 ParentId는 송신자

가 속한 IG, 닻, 그리고 부모를 나타낸다. 송신자가 자식을 더 이상 받아들일 수 없는 경우에 OverCapacity를 1로 셋하여 보낸다. HopsToAnchor는 송신자에서 닻까지의 홉 수 이다. IG는 관리하고 있는 닻 노드들 중의 하나를 선정하여 IG-Hello를 주기적으로 유니케스트한다. 단, 관리하고 있는 닻 노드가 없는 경우에만 브로드케스트한다. IG-Hello = (MsgId, IGId, OverCapacity)로 정의된다. IGId는 송신자의 고유번호이다. OverCapacity는 IG가 관리하는 닻의 수가 한계를 초과하여 더 이상 닻의 가입을 수용할 수 없는 경우에 1로 셋 된다.

▪**AREQ, ARES**: 이동 노드는 새로운 닻이 되기 위하여 자신의 이웃 정보 테이블에 있는 IG 목록에서 OverCapacity가 0인 특정 IG를 선택하여 닻 요청 메시지 AREQ(Anchor Request)를 유니케스트한다. AREQ = (MsgId, SrcId, DstId, UrgentFlag, IGId)로 정의된다. 부모나 혹은 기존의 IG에 대한 연결이 끊어져 긴급하게 새로운 IG를 찾는 고아 노드는 OverCapacity가 0인 IG가 없으면 임의의 IG를 선택하여 UrgentFlag를 1로 셋하여 보낸다. 하지만, 고아 상태에 있었던 OverCapacity가 0인 IG가 자신의 목록에 없으면 AREQ를 보내지 않는다. UrgentFlag가 1인 경우에는 수신 IG는 수용한도가 넘어선 경우에도 요청 노드를 닻으로 받아들인다. 수신 IG가 새로운 닻을 허용하면 응답으로 ARES (Anchor Response)를 보낸다. ARES = (MsgId, SrcId, DstId, AcceptedFlag, IGId)로 정의되고 새로운 닻을 허용하면 AcceptedFlag = 1로 셋 된다.

▪**JREQ, JRES**: 멤버가 되기 위한 고아 노드는 자신의 이웃 정보 테이블에 있는 OverCapacity가 0인 특정 이웃 노드를 선택하여 JREQ (Join Request)를 특정 노드에게 유니케스트한다. 만일 가입 수락 메시지 JRES (Join Response)를 받지 못하면, 약간의 쉼(Interval) 후에 OverCpacity가 0인 다른 이웃 멤버 노드를 선택하여 다시 보낸다. JREQ = (MsgId, SrcId, DstId, UrgentFlag, IGId) 이며, JRES = (MsgId, SrcId, DstId, AcceptedFlag, IGID, AnchorId, HopsToAnchor)로 정의된다. UrgentFlag와 AcceptedFlag의 사용 방법은 AREQ 및 ARES의 그것들과 동일하다.

▪**PCREQ, PCRES**: 부모 노드의 변경을 원하는 멤버 노드는 PCREQ (Parent Change Request) 를 OverCapacity가 0을 가진 새로 선정한 이웃 노드에 보낸다. PCREQ = (MsgId, SrcId, DstId, IGId)로 정의된다. PCREQ를받는 노드는 응답 메시지 PCRES (Parent Change Response)메시지를 보낸다. PCRES = (MsgId, SrcId, DstId, IGId, AnchorId, HopsToAnchor, AcceptedFlag)로 정의된다. PCREQ를 수신한 노드가 자식을 허용한 경우에는 AcceptedFlag 1[1] 로 셋 된다.

▪**UREQ**: 멤버 노드는 토폴로지의 변경을 통보하기 위하여 갱신 요청 메시지 UREQ(Update Request)를 보낸다. UREQ = (MsgId, SrcId, DstId, DirFlag, Action, [IGId, AnchorId, MemberList])로 정의된다. 트리의 루트를 향해서 전달되는 UREQ의 DirFlag는 하위 방향이면 0, 상위 방향이면 1, 양방향이면 2로 셋 된다. DirFlag = 0(하위 방향)인 경우에는 Action = ORPHAN으로 셋되며 송신자가 고아가 되고 수신자인 자식 노드는 부모의 링크를 제거해야 한다는 것을 나타낸다. DirFlag = 1(상위 방향)인 경우에는 Action = ADD 혹은 DELETE가 되고, 이것은 부모 노드가 MemberList에 포함된 노드 목록을 추가 혹은 삭제할 것인지를 나타내며, DirFlag = 2(양방향)인 경우에는 Action = UPDATE가 되며, 수신하는 모든 노드가 IGId, AnchorId, 및 MemberList에 따라 정보를 갱신할 것을 나타낸다.

Ⅲ. 이동성 관리 프로토콜

3.1 이웃 관리

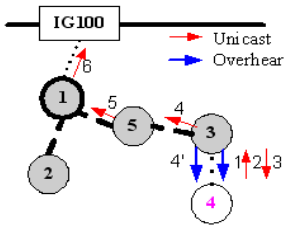
▪IG는 Hello-Interval마다 주기적으로 자신이 관리하는 닻들 중의 하나를 선정하여 IG-Hello를 유니케스트한다. 이 경우에 일정 시간 동안 MAC 계층으로부터 ACK 메시지를 받지 못하면 IG-Hello가 전송되지 않은 것으로 간주하고 정해진 시간 간격, IG-Hello-Retry-Interval이 지난 후에 다른 닻 노드를 선정하여 재전송한다.

▪멤버(닻)은 자신의 부모(IG)에게 Hello-Interval마다 주기적으로 MN-Hello를 전송한다. 멤버 혹은 닻 노드는 ACK 수신을 통해서 자신의 상위 링크가 유효한 상태인지를 즉시 알 수 있다. 이 과정에서 수신 혹은 엿듣는 모든 노드는 자신의 NeighborTable을 갱신한다.

3.2 닻 요청 절차

▪고아 노드는 IG-Hello를 엿듣는 즉시 송신한 IG

1) 노드가 관리하고 있는 이웃 노드의 OverCapacity정보는 Hello메시지의 전송간격 때문에 실제 해당 노드의 OverCapacity 정보와 다를 수 있다. 따라서, 부모 변경 요청이 실패할 수 있다.



1. Orphan 4 overhears MN-Hello with OverCapacity = 0 form node 3
2. Send JREQ to node 3
3. Receive JRES with AcceptedFlag = 1 from node 3
4. Send UREQ with DirFlag = 2, Action = UPDATE, IGId, AnchorId, and MemberList = {4}
4'. Node 4 overhears UREQ
5. Node 5 sends UREQ with DirFlag = 1, Action = Add, MemberList = {4}
6. Node 1 sends the same as 5 to IG100.

그림 5. 가입 절차

의 OverCapacity = 0이면 해당 IG의 닻이 되기 위하여 AREQ 메시지를 보낸다.

▪수신 IG는 새로운 닻을 수용할 수 있으면 ARES 메시지로 응답한다.

▪ARES를 수신한 고아 노드는 Accepted Flag = 1이면 닻이 되고, 기 수신한 IG-Hello와 수신한 ARES에 포함된 정보를 이용하여 자신 만의 TreeInformationTable을 작성한다.

▪IG는 송신한 ARES에 대한 ACK (MAC 계층)를 수신하면 새로운 닻을 등록한다.

그림 2는 고아 노드 4가 멤버 3에 가입하는 경우에 프로토콜 작동 과정을 도시한 것이다. 노드 3은 노드 4의 가입 요청 메시지 JREQ를 받은 후에 JRES를 보내고 MAC 계층으로부터 ACK를 받으면 정보 갱신을 시작한다. 노드 3은 DirFlag를 양방향으로 셋하고 IGId, AnchorId, 새로운 멤버 4를 포함하는 UREQ를 노드 5에게 보낸다. 이 때 새로운 노드 4는 엿듣기를 통해서 IGId와 AnchorId을 갱신한다. 노드 5은 UREQ에서 IGId와 AnchorId를 삭제하고 DirFlag를 1(상위 방향)로 변경하고 Action을 ADD로 변경하여 노드 1에게 송신한다. 노드 1은 새로운 동일한 메시지를 계속해서 IG100에게 전달한다.

3.3 부모변경 절차

▪멤버 x가 이웃 y로부터 OverCapacity = 0인 MN-Hello를 받으면, x의 닻까지의 거리와 y의 닻까지의 거리에 1을 더한 값을 비교한다. 그 차이가 T_{change}보다 크거나 같으면 x는 자신의 부모 노드의 변경을 결정하고 y에게 PCREQ를 보낸다.

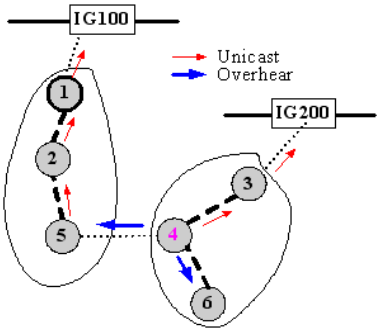


그림 6. 부모 변경 (노드 4의 부모가 5에서 3으로 변경)시 UREQ 전송

▪x가 AcceptedFlag = 1인 PCRES를 받으면 자신의 트리 정보 (부모, 닻, IG)를 갱신하고, 갱신된 정보와 자신을 루트로 하는 서버 트리의 정보를 포함하는 UREQ를 새로운 부모 노드에게 보낸다. 부모 노드는 수신한 UREQ 메시지가 가지고 있는 MemberList만을 포함하는 UREQ를 상위 노드로 전송하며 모든 선조 노드 및 IG는 트리 정보를 갱신한다.

▪UREQ를 수신한 자식 노드는 변경된 닻과 IG만을 포함하는 UREQ를 계속해서 자식노드로 전달한다. 결국 모든 후손 노드는 새로운 닻과 IG로 갱신한다.

▪이전의 부모 노드는 MemberList를 취하여 계속해서 상위 노드로 전달하고 모든 이전의 선조 노드들과 이전 IG는 MemberList를 참조하여 정보를 삭제한다. 여기서 T_{change}를 1로 설정하면 모든 노드는 IG까지 최단 거리를 유지하려고 할 것이다. T_{change}값이 크면 패킷 전송 오버헤드는 늘어날 것이고 부모 변경 프로토콜로 인한 오버헤드는 줄어들 것이다.

그림 3은 노드 4가 부모를 노드 5에서 노드 3으로 변경하는 경우에 정보 갱신 과정을 보여준다. 노드 4는 DirFlag = 2로 셋한 후에 UREQ 메시지를 새로운 부모인 노드 3에게 유니케스트한다. 이 메시지를 받은 노드 3과 IG200는 노드 4와 6을 추가한다. 동일한 UREQ 메시지를 엿들은 노드 6은 새로운 것으로 자신의 IG와 닻을 갱신한다. 동일한 메시지를 엿들은 이전 부모 5는 자신의 정보 테이블에서 4와 6을 삭제하고, DirFlag = 1, Action = DELETE로 셋하여 위로 보낸다. 선조 노드 2, 1, 그리고 IG100은 노드 4와 6을 제거한다. 이런 식으로 노드 4는 하나의 UREQ를 전송하여 토폴로지 갱신을 완료한다.

3.4 링크 파손 복구절차

파워 소진, 시스템 오류, 노드 이동 등으로 인하여 트리의 링크가 끊어질 수 있다. 링크의 끊어짐을 신속하고 효율적으로 감지하는 것은 토폴로지의 정확성을 유지하는데 있어서 매우 중요하다.

▪ 노드는 하위 링크에 대하여 지정된 시간 간격 즉, Hello Interval+ϵ (ϵ << HelloInterval) 동안 어떤 자식 노드로부터 MN-Hello를 받지 못하는 경우에 그 링크는 끊어진 것으로 간주한다[2]. 노드는 MN-Hello를 자신의 부모 노드에 보낼 때 마다 ACK를 수신함으로써 상위 링크의 연결 상태를 확인한다. 부모 노드가 없는 닻 노드의 경우에는 IG에게 MN-Hello를 송신한다. ACK가 없는 경우에 상위 링크는 즉시 끊어진 것으로 간주한다. 따라서, 상위 및 하위 링크 연결 파손의 최악의 감지시간은 HelloInterval + ϵ이 된다.

▪ IG의 경우에는 주기적으로 자신이 관리하고 있는 닻 노드들 중의 하나를 선택하여 IG-Hello를 보내고 다른 노드들은 엿듣기를 한다. 선택한 닻 노드에 대한 링크가 끊어져서 IG-Hello를 전송하지 못하거나[3] 혹은 전송을 한 후에 ACK를 수신하지 못하는 경우가 발생할 수 있다. 전자의 경우에는 일정 시간 동안 ACK를 받지 못하면 다른 닻 노드를 선택하여 IG-Hello를 보낸다. 물론, 관리하고 있는 닻 노드가 없는 경우에는 IG-Hello를 브로드캐스트 한다.

▪ 하위 링크 끊김을 감지한 노드는 UREQ를 그 자신의 조상 노드들 및 IG에게 보낸다. 상위 링크 끊김을 감지한 노드는 임시 루트 (root) 노드 (후손 노드를 갖고 있지만 자신은 pMANET의 멤버가 아닌 노드) 가 되고 자신의 NeighborTable을 참조하여 새로운 IG의 닻이 되거나 혹은 다른 멤버 노드에 가입한다. 이 때 OverCapacity가 0인 아닌 노드가 없는 경우에는 OverCapacity = 1인 노드를 선정할 수 있다. 링크 파손을 복구해야 하는 긴급한 상황이므로 UrgentFlag는 1로 설정한다. 먼저 가입할 IG를 찾고, 없으면 가입할 이웃 멤버 노드를 찾는다. 만일 성공한다면 UREQ를 자신의 선조 및 후손

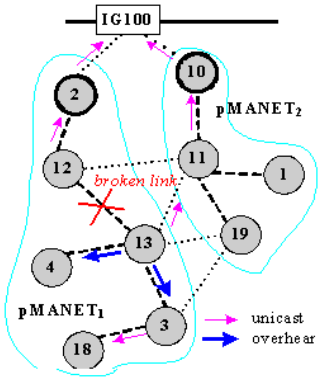


그림 7. 트리 링크 파손 복구 절차

들에게 보낸다. 이 경우에도 UREQ의 전송 방향을 양방향으로 설정하고 방금 가입한 IG 혹은 부모에게 유니케스트하고다른 노드들은 엿듣기을 통해서 수신한다. 성공하지 못한 경우에는 UREQ 메시지를 자신의 자식 노드들 중의 하나에게 보내고 자신은 다른 MN-Hello 혹은 IG-Hello를 받을 때까지 고아 노드가 된다. UREQ를 수신하는 각 자식 노드는 그 자신의 부모와 동일한 과정을 통해서 닻, 멤버 혹은 고아가 된다.

그림 4에서 $pMANET_1$의 링크 (12, 13)이 끊어졌다고 가정하자. 하위 링크 끊어짐을 감지한 노드 12는 UREQ를 조상 노드 2와 IG100에게 보낸다. 상위 링크 끊어짐을 감지한 노드 13은 NeighborTable를 참조하여 멤버 노드 11이 이웃임을 알게 되고, 긴급 가입을 수행한다. 가입이 성공하면 UREQ의 DirFlag를 2 (양방향)로 설정하고 부모 노드 11에게 보내고노드 11은 수신한 UREQ의 DirFlag를 1(상방향)로 변경하여 노드 10에게 전달하고, 결국 UREQ는 IG100에게 전달된다. 또한, 자식 노드 3, 4는 동일한 UREQ를 엿듣고, 방향 플래그를 체크한다. DirFlag = 2이기 때문에 UREQ를 수신한 노드 3은 DirFlag를 0으로 변경하여 노드 18에게 전달한다.

Ⅳ. TBRP 라우팅 프로토콜

트리 기반의 통합 네트워크 아키텍처에 적합한 트리 기반의 라우팅 프로토콜을 경로 설정, 개선 및 복구로 나누어 설명한다. 경로 설정 방식은 소스(src)와 목적지(dst)의 상대적인 위치에 따라 다르다.

1. src가 인터넷에 있고 dst가 MANET에 있는 경우
   이 경우에 경로는 두 경로 세그먼트의 조합으로

2) MN-Hello 메시지를 보낼 때 브로드케스트를 사용하지 않기 때문에 링크가 끊어져 있지 않으면 자식 노드는 자신의 부모 노드에게 HelloInterval마다 성공적으로 MN-Hello를 보낸다고 가정한다. ϵ은 MN-Hello를 보낼 때 전송 지연을 고려하기 위한 값이다.

3) 정보 테이블에서 선택한 닻 노드가 움직여서 MAC 계층에서 RTS-CTS 교환이 실패한 경우, 즉 채널 확보를 실패한 경우에 IG-Hello를 보낼 수 없다.

이루어 진다: (src, …, IG) 및 (IG, …, dst). 첫 번째 세그먼트는 인터넷 호스트에서 목적지 노드가 등록되어 있는 IG까지의 경로로써 모든 이동 노드가 이미 FA 및 HA에 가입되어 있다고 할 때 Mobile IP에 의해서 처리된다. 두 번째 세그먼트는 모든 트리의 각 노드는 자신의 각 후손으로 인도되는 다음 자식 노드 정보를 각 노드의 TreeInformationTable이 가지고 있기 때문에 경로 설정이 필요하지 않다.

2. src가 MANET에 있고 dst가 MANET 혹은 인터넷에 있는 경우

이 경우에 경로는 두 경로 세그먼트의 조합으로 표현된다: (src, …, IG) 및 (IG, …, dst). 첫 번째 세그먼트는 트리 경로를 따라서 이미 설정되어 있으며 두 번째 세그먼트는 dst가 MANET에 있는 경우에는 Mobile IP가 담당하고 인터넷에 있는 경우는 글로벌 IP프로토콜이 담당한다.

위에서와 같이 단순 경로를 사용하는 경우에 이동 노드 수의 증가와 비례하여 인프라구조의 오버헤드가 크게 증가할 수 있다. 따라서, 기 설정한 설계목표를 충족시키기 위해서 이동 노드들끼리의 통신은 되도록 MANET 자원 만을 사용할 수 있도록 경로설정을 하는 것이 바람직하다. 그렇지만, 해당 이동 노드 사이의 통신 효율성을 고려하지 않는다면 그 이득은 상쇄될 수 있다. 즉, 통신을 원하는 두 개의 이동성 노드가 비교적 멀리 떨어져 있는 경우에는 MANET 자원만을 사용하여 통신을 하는 것은 통신 성능을 저하시킬 수 있다. 따라서, 양 통신 당사자의 위치에 따라서 경로설정을 달리 함으로써 MANET 자원의 효율적인 이용과 성능 향상이라는 두 개의 직교하는 설계 목표를 어느 정도 성취할 수 있다.

src와 dst의 위치에 따라서 다음과 같이 네 개의 통신 유형을 정의한다.

- Type1: dst가 src의 후손이거나 혹은 이웃이다.
- Type2: src와 dst가 동일한 pMANET에 속 하지만 dst가 src의 후손이 아니다. 즉, dst는 src의 선조이거나 혹은 자신의 형제 (선조의 후손)이다.
- Type3: src와 dst가 게이트웨이가 동일한 두 개의 다른 pMANET에 속한다.
- Type4: src는 pMANET에 속하고 dst는 인터넷이나 혹은 src와 다른 게이트웨이를 사용하는 어떤 pMANET에 속한다.

4.1 경로 설정

경로 설정을 원하는 노드는 목적지가 자신의 후손인지 혹은 이웃인지를 알아 보기 위해서 먼저 자신의 TreeInformationTable과 NeighborTable을 참조한다. Type1 경로는 이들 테이블 정보를 통해서 이미 최단 거리로 설정되어 있으며 라우트 탐색이 필요 없다. 이 경우 path = (src, …, dst)이 되고 경로상의 모든 노드는 이동 노드가 된다.

Type1이 아니면 src는 Route Request (RREQ)를 상위 트리 경로를 따라 전송한다. RREQ가 이동하는 과정에서 RREQ를 수신하는 모든 선조 노드들은 dst가 자신인지 혹은 자신의 후손인지를 확인한다. 그렇다면, 해당 노드는 Route Reply (RREP)를 src로 보낸다. 따라서, Type2 경로는 (src, …, co-ancestor)와 (co-ancestor, …, dst)의 조합이며 co-ancestor는 src 및 dst의 공통 선조이다.

Type1 경로와 Type2 경로는 모두 MANET 자원만을 사용한다. 지역 통신을 목적으로 하는 MANET의 특성상 Type1 혹은 Type2의 가능성을 높다고 할 수 있다. 특히, pMANET의 크기가 클수록 그 가능성은 증가한다.

만일 RREQ가 src.IG에 도달할 때 까지 dst를 찾지 못하고, src.IG는 dst가 자신의 관리하고 있는 pMANET 중의 하나에 속하는지 검사한다. 그렇다면, Type3 경로 (src, …, src.anchor, src.IG, dst.anchor, …, dst)가 설정된다. src.IG는 이 경로를 포함하는 RREP를 src에 보낸다.

만일 src.IG가 해당 노드를 자신이 관리하는 pMANET들 중에서 찾을 수 없는 경우에는 부분 경로 (src, …, src.anchor, src.IG)를 포함하는 RREP를 src에 보낸다. 이 경우 Type4 경로가 되고, src가 보내는 모든 패킷은 src.IG로 가서 목적지가 유선 호스트이면 글로벌 IP, 다른 MANET에 속하면 글로벌 IP 및 Mobile IP의 도움으로 전송을 완료한다.

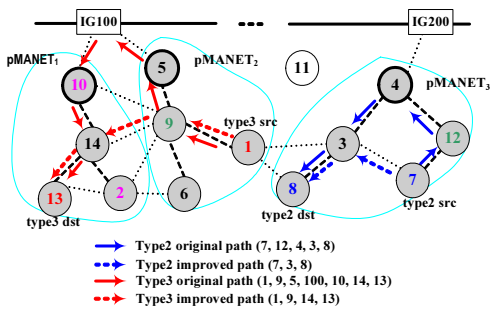


그림 8. 경로 유형에 따른 경로 개선

4.2 경로 개선

RREQ가 위쪽으로 전송되는 동안에 엿듣는 노드는 dst가 자신의 후손이거나 이웃이면 src에 RREP 응답 메시지를 보낸다. 그림 5에서 Type3의 경로 유형인 (src, dst) = (1, 13)인 경우를 보자. 노드 10, 14, 2는 9가 5로 전달하는 RREQ를 엿들을 수 있다. 노드 2은 13이 이웃이라는 것을 알고, 10과 14는 둘 다 13이 자신의 후손이라는 것을 알 수 있다. 따라서, 그들은 모두 9에게 dst까지 거리를 포함하는 RREP를 직접 보낸다. 그들이 RREP를 동시에 보내는 경우에 충돌이 발생할 수 있다. 다음과 같이 grace-delay d 함수[2]를 사용하여 약간의 임의 지연 시간을 가지고 RREP를 응답한다:

$$d = H \times (h - 1 + r) \qquad (1)$$

여기서 h는 RREQ를 송신한 노드에서 목적지까지 홉 수, r은 0과 1사이의 임의의 값이고, H는 홉 간 전송 지연을 나타내는 상수 값이다. d값은 목적지까지 더 짧은 거리를 제공하는 노드에게 RREP를 전송할 우선권을 준다. 따라서, 원래의 Type3 경로인 (1, 9, 5, 100, 10, 14, 13) 대신에 단축된 경로 (1, 9, 14, 13) 혹은 (1, 9, 2, 13)가 설정된다.

또한, 소스 노드가 6이고 목적지가 13인 경우를 보자. 이 경우는 두 개의 노드가 공통의 이웃 2을 가지고 있다. 이 경우 역시 Type3경로를 가져야 하지만, 노드 6이 노드 9로 보내는 RREQ를 노드 2가 엿듣게 되고 노드 2는 즉시 RREP를 보낸다.

4.3 경로 복구

전송 중에 파손된 경로 복구에 대하여 다음 네 가지 경로유형에 따라 설명한다.

▪ (*Type1 경로 파손 복구*) Type1의 어떤 링크(x, y)가 파손되었다고 가정해 보자. 이 경우 x가 패킷을 y에게 전송하는 즉시 x는 ACK (MAC 계층)의 수신을 체크함으로써 전송 성공을 확인한다; ACK를 수신하지 못하면 링크가 끊어진 것으로 간주한다. x는 버퍼에 패킷을 저장하고, y를 찾기 위해서 TTL = 2 (y는 2홉 내에 있을 확률이 높음)인 RREQ를 보낸다. 만일 (y, …, dst)상의 한 노드가 RREP를 x에게 응답하면 경로 복구는 완료되고, 버퍼에 저장된 패킷을 전송 완료한다. 만일 x가 경로를 복구하지 못하면 경로 파손 에러 메시지 RERR (Route Error)을 src에게 보낸다.[4] src는 약간의 간격 즉, 파손된 토폴로지가 트리 링크 파손 복구 절차에 의한 링크 복구 시간, *Link-Recovery-Time*, 을 기다린 후에 경로 복구를 재시작 한다. src는 새로운 경로를 설정한 후에 재전송을 시작한다.

▪ (*Type2 경로 파손 복구*) 경로 세그먼트 (src, …, co-ancestor)상의 어떤 링크 (x, y)가 끊어진 경우를 보자. 노드x는 Type1의 경로 복구에서처럼 링크가 끊어진 것으로 판단되면 즉각 보유중인 패킷을 버퍼에 저장하고 y를 찾기 위해서 TTL = 2의 RREQ를 플러딩 한다. x가 (y,.., co-ancestor, …, dst)중의 한 노드로부터 RREP를 수신하면 x는 경로 복구는 완료된다; 경로를 찾지 못한 경우에는 RRER을 src에 보낸다. RRER을 수신한 src는 약간의 간격 즉, *Link-Recovery-Time* 동안 기다린 후에 경로 재설정을 수행한다. 부분 경로 (co-ancestor, …, dst)상의 어떤 (x, y)가 끊어지면 링크 복구 과정은 Type1과 유사하다. 링크 복구가 실패하면 src에 RRER을 보내서 경로 재설정을 요청한다.

▪ (*Type3 경로 파손 복구*) 부분 경로 (src, …, src.anchor)상의 링크(x, y)가 끊어지면 복구 과정은 Type2의 경우와 같으며, 부분 경로 (dst.anchor, …, dst)상의 (x, y)가 끊어지면 복구 과정은 Type1의 경우와 같다. 끊어진 링크 (x, y)가 (src.anchor, IG)이면 x(= src.anhcor)는 TTL = 2를 갖는 RREQ를 보내고 dst까지 경로를 알고 있는 IG나 혹은 어떤 노드가 RREP를 응답할 것을 기다린다. 응답 노드들은 grace-delay 함수를 사용한다. x가 오직 다른 IG으로만 RREP를 받는다면 그 IG로 경로를 만든다. x가 이동 노드로부터 RREP를 받는 경우에는 x는 그 이동 노드로 경로를 만든다. 끊어진 링크 (x, y)가 (IG, dst.anchor)라면, x는 TTL =2인 RREQ를 전송하여 (dst.anchor, …, dst)중의 어떤 노드, 즉 dst를 후손으로 가지고 있는 어떤 노드가 RREP를 응답하는 것을 기다린다. 응답 노드들은 grace-delay 함수를 사용한다. 응답하는 노드가 없는 경우에는 IG는 즉시 src로 RERR를 보낸다.

▪ (*Type4 경로 파손 복구*) 부분 경로(src, …, src.anchor, src.IG)상의 어떤 링크 (x, y)가 끊어진다고 하자. x가 src.anchor라면 TTL = 1인 RREQ를 보내고 임의의 IG까지 가장 짧은 경로를 제공하는 노드 (IG 포함)를 찾는다. x가 다른 이동 노드들 중의 하나라면 Type2 링크 복구 과정과 같다.

4) RRER을 먼저 src에 보내고 복구를 수행할 수 있으나 TTL 값을 2로 하고 지연 시간 없이 복구를 수행하기 때문에 경로 복구를 먼저 수행해도 문제가 없다.

V. 성능 평가

GlomoSim의 상용 버전인 QualNet 3.9 네트워크 시뮬레이터를 사용하였다. 정사각형 영역에서 100개의 이동 노드 3개의 IG를 사용였으며, 이동 노드는 무작위로 분표되고 고정 IG인 경우에 실제 환경에서 기지국처럼 인위적으로 배치되기 때문에 미리 정해진 위치를 가정한다. 사용된 파라메타 값들은 표 2와 같으며, 4개의 시나리오 그림 6-(a), 6-(b), 6-(c), 6 - (d)는 동일한 영역 1000 x 1000(m^2)에서 서로 다른 IG 배치 패턴을 가지고 있다. 다섯 번째 시나리오는 시나리오 4-(a)의 IG배치와 같지만 영역 1500 x 1500(m^2)을 사용하였다.

시나리오들의 특징을 살펴보기로 하자. 기본적으로 배치된 IG가 더 넓은 영역을 커버하면 할수록 더 많은 닿 노드들을 가질 수 있기 때문에 트리의 수는 많아지며, 각 트리의 크기가 작아지고, 멤버 노드들이 더 짧은 거리에 IG를 엑세스할 수 있다. 4방향이 개방되도록 놓인 IG에 비해서 귀퉁이에 있는 IG는 1/4 영역만을 커버하고 가장자리에 있는 IG는 1/2영역만을 커버하기 때문에 시나리오 4-(a) 가장 좋은 반면 4-(d)는 가장 좋지않은 배치라고 할 수 있다.

표 2. 시뮬레이션 파라메타

Parameters	Value
Mobility Pattern	Random Waypoint
Pause Time	0
Number of Nodes	103 (3 fixed IGs included)
Dimension	1000 x 1000, 1500 x 1500
Transmission Range	250 m
Wireless Bandwidth	2 Mbps
Traffic Pattern	CBR
Number of Sessions	15 (1 packet/sec.)
Packet Size	512 bytes
Simulation Time	600 seconds

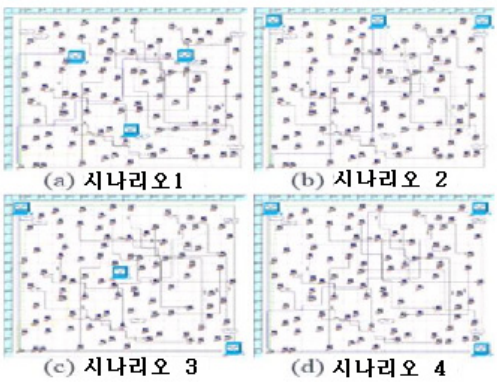


그림 6. 통합 네트워크의 4가지 전개 시나리오

5.1 시나리오 분석

그래프 표시에서 영역 k을 갖는 시나리오 n은 $S_{n\text{-}k}$로 표시하며, n = 1…4 그리고 k = 1000 x 1000 혹은 1500 x 1500이다.

그림 7과 8을 보자. 전반적으로 속도가 높아짐에 따라 트리의 수가 약간씩 감소하는 패턴을 볼 수 있다. 이것은 순간적으로 트리에 속하지 않는 노드의 수가 증가하기 때문이다. 또한, IG들이 커버하는 영역이 가장 큰 시나리오 1은 가장 많은 수의 트리를 가지며, 그 결과 평균 트리의 크기가 가장 작음을 알 수 있다. 3개 IG의 전체 커버 영역이 가장 작은 시나리오 4는 기대한 대로 트리의 수는 가장 적고 평균 트리의 크기는 가장 크다. IG 커버 영역

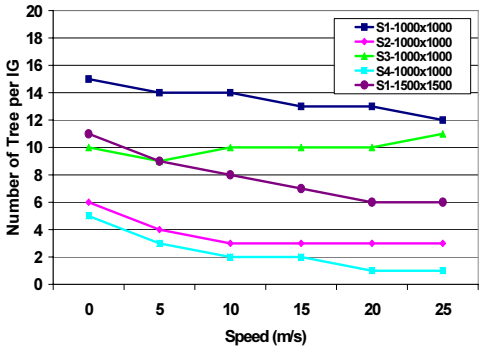


그림 7. 속도 변경에 따른 IG당 트리의 수

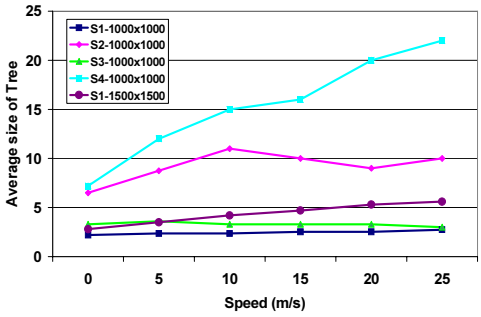


그림 8. 속도 변경에 따른 평균 트리의 크기

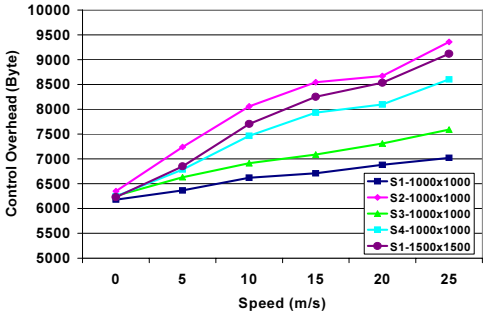


그림 9. 속도 변경에 따른 콘트롤 오버헤드

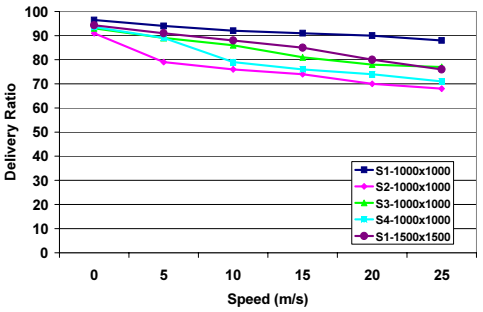


그림 10. 속도 변경에 따른 전송율

에 비례하여 평균 트리의 수는 S1-S3-S2-S4와 같으며, 평균 트리의 크기는 그 역순이 된다.

이러한 현상은 그림 9의 네트워크 오버헤드와 관련성을 갖는다. 즉, 일반적으로 평균 트리의 크기가 클수록 오버헤드는 크다. 하지만, 시나리오 2와 4의 경우에는 트리의 수는 시나리오 2의 경우가 더 많고 평균 트리의 크기는 작지만, 오버헤드는 시나리오 4보다 큰 것으로 나타났다. 시나리오 2의 경우에 그 배치의 특성상 많은 닻들이 중간 가장자리에 위치한 IG에 속할 확률이 높고, 중간에 있는 IG에 소속된 pMANET가 비교적 큰 트리가 된다. 따라서, 전체적으로 오버헤드가 S4에 비해 크다. 그림 10을 보면 전송율은 오버헤드와 역으로 나타남을 알 수 있다.

시나리오 5의 경우에 낮은 노드 밀도로 인하여 닻 노드의 수가 적고, 그 결과 트리의 수가 거의 배로 감소하고 크기는 거의 2배 만큼 커졌다 (그림 8). 또한, 노드 사이의 거리가 증가로 인한 빈번한 링크 파손으로 복구 오버헤드가 증가하여 시나리오 1에 비해 오버헤드가 크게 증가하였다 (그림 9).

그림 11은 이동 노드들 간의 통신에서 어느 정도 인프라구조 자원이 사용되는 가를 보여준다. 노드의 수가 과다하게 증가하면 인프라구조 네트워크상에 부가되는 오버헤드가 증가하고, 그 결과 전체 네트

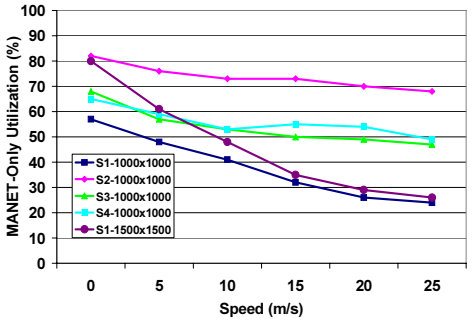


그림 11. 패킷 전송을 위한 MANET-Only 자원의 이용률

워크 효율이 감소할 수 있다. 시뮬레이션 결과를 보면 작은 크기의 트리를 갖는 시나리오 1에서 조차도 노드 속도에 따라 통신의 25% 내지 58%가 MANET 자원만을 사용한다는 것을 알 수 있다. 시나리오 2에서 이 비율이 가장 크게 나타나는 이유는 중간 노드의 트리가 커지는 현상 때문에 Type1 경로 혹은 Type 2 경로가 설정될 가능성이 높아지기 때문이다. 하지만, 시나리오 4의 경우에 세 개의 IG가 모두 코너에 있기 때문에 평균 사이즈가 비슷한 트리들이 형성되어 결과적으로 MANET 자원만의 이용률이 시나리오 2의 그것보다 낮다. 전반적으로 전송률이 높음에도 불구하고 MANET 자원만을 사용하는 통신 비율이 높은 현상은 인프라구조 자원을 덜 사용하고자 하는 우리의 목표에 부합한다. 또한, 노드 이동 속도가 높아짐에 따라 MANET가 불안정해 지기 때문에 인프라구조 자원을 사용하는 비율이 높아진다는 것을 확인할 수 있다.

시뮬레이션 결과에 따르면 시나리오 1이 가장 최선의 IG 배치라고 할 수 있다. 시나리오 5의 경우에 시나리오 1에 비해서 트리 수의 감소로 인한 평균 트리의 크기가 커지고, 오버헤드가 증가한다. 하지만, 두 시나리오 사이의 전송률의 차이는 그림 10에서처럼 중간 속도 (5m/s)까지는 그렇게 차이가 나지 않는다. 속도가 높아지면 노드 사이의 결합력이 약한 시나리오 5에서의 트리가 불안정성이 높아지고 따라서 전송률의 감소 속도가 약간 빠름을 알 수 있다. 그림 11을 보면 평균 트리 사이즈가 크기 때문에 경로 Type1 및 Type2를 가질 확률이 높은 시나리오 5가 시나리오 1에 비교해서 MANET-Only 자원의 이용률이 높다는 것을 알 수 있다.

5.2 기존 프로토콜과의 성능 비교

성능 비교를 하기 위해서 가장 잘 알려진 AODV를 MANET 라우팅 프로토콜로 선정하였다[5]. 효율

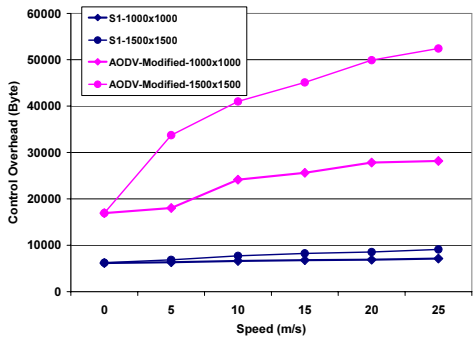


그림 12. 속도 변화에 따른 콘트롤 오버헤드

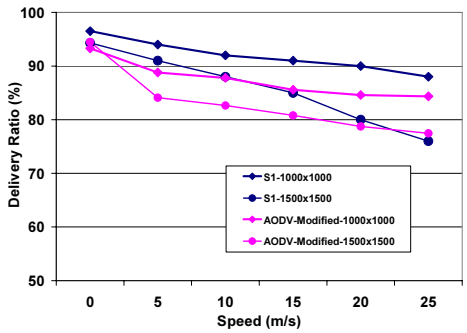


그림 13. 속도 변화에 따른 전송율

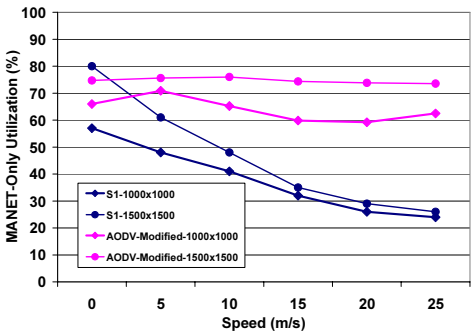


그림 14. MANET-Only 자원의 이용율

적인 노드 이동성 관리를 위하여 사전적 방식과 반응적 방식을 조합한 하이브리드 방식을 사용하였다. 이동 노드의 등록을 위해서 IG는 K_1홉까지 에이전트 광고 (Agent Advertisement) 주기적으로 메시지를 플러딩한다. 에이전트 광고 메시지가 플러딩되는 동안에 수신 노드는 IG까지 역경로를 설정 혹은 갱신한다. 에이전트 광고 메시지를 수신하는 등록되지 않은 노드는 역경로를 따라 등록 요청 메시지를 송신 IG에 보낸다. 이 과정에서 IG로부터 수신 노드로까지 순방향 경로가 설정된다. 등록 요청 메시지를 받은 IG는 해당 이동 노드를 FA 및 HA에 등록 완료한다. 등록이 완료된 이동 노드는 타임 아웃이 발생할 때까지 IG까지 양방향 경로를 유지한다.

에이전트 광고 메시지를 수신하지 못하는 K_1보다 멀리 떨어진 노드는 통신을 원하는 경우에 TTL = K_2를 갖는 에이전트 요청 (Agent Solicitation) 메시지를 플러딩하여 이미 등록된 가장 가까운 노드를 찾는다. 이 과정에서 기 등록된 노드로부터 에이전트 요청 메시지를 보낸 노드로까지 순방향 경로가 설정된다. 에이전트 요청 메시지를 받는 기 등록된 노드는 응답 메시지를 송신하고 이 응답 메시지가 이동하는 동안 기 등록된 노드까지 역방향 경로가 설정된다. 따라서, 통신을 원하는 노드는 자신으로부터 기 등록된 노드까지, 기등록된 노드로부터 IG까지 설정되어 있는 순방향 경로를 통해서 등록 요청 메시지를 보냄으로써 FA 및 HA에 등록한다.

어떤 노드가 통신을 원하는 경우에는 RREQ (TTL = K_3)를 전송하고 IG 혹은 목적지 노드는 메시지를 받는 즉시 RRES를 응답한다. 목적지 노드가 응답한 경우에는 MANET 자원만을 사용하여 직접 통신을 한다. IG가 응답한 경우에는 응답한 IG들 중에서 가장 가까운 IG를 선택하고 해당 IG를 이용하여 목적지 노드와 통신을 수행한다.

두 방법을 비교하기 위하여 가장 최선의 IG 분표를 가진 것으로 평가된 시나리오 1과 그것의 확장된 영역을 갖는 시나리오 5를 사용하였다. 트리 기반 라우팅 프로토콜의 경우 2초 간격으로 송신하는 Hello 메시지, 트리 기반 통합 네트워크 구조 유지를 위한 콘트롤 메시지, 그리고 라우트 설정을 위한 메시지들을 모두 포함하였다. 그림 12를 보면 트리 기반 라우팅 방식이 AODV를 사용하는 기존의 방식보다 훨씬 오버헤드가 적다는 것을 알 수 있다. 그 결과 5 ~ 10% 정도 전송률 개선을 보인다. 하지만, MANET 자원만의 이용률은 AODV가 훨씬 높다는 것을 알 수 있다 (그림 14). AODV의 경우에 목적지 노드에 대한 경로를 찾기 위해서 RREQ를 플러딩하기 때문이다. 속도가 낮은 경우에는 트리 기반 라우팅 방식의 MANET 자원 이용률이 AODV방식의 그것과 크게 차이가 나지 않는 다는 것을 알 수 있다. 노드 이동성이 높으면 순간적으로 트리 구조를 깨뜨려 노드들이 IG에 더 많이 의존하게 되기 때문이다.

Ⅵ. 결 론

이동 ad hoc 네트워크에서 Mobile IP를 효율적

으로 지원하기 위한 트리 기반 통합 네트워크를 제안하고, 통합 네트워크의 형성 및 관리를 위한 네트워크 구조 관리 프로토콜, 그리고 통합 네트워크에 특화된 라우팅 프로토콜인 트리 기반 라우팅 프로토콜을 제안하였다. 이동성 노드는 플러딩을 사용하지 않고 트리 구조를 따라 쉽게 FA 및 HA에 등록한다. 멤버 노드가 IG까지 최단 거리를 유지하는 식으로 트리가 형성되기 때문에 인터넷 엑세스 효율성이 높다. 또한, 제안된 라우팅 프로토콜은 각 노드의 트리 정보를 이용하여 플러딩을 사용하지 않고 효율적으로 경로를 설정한다. 주어진 시나리오에 대하여 수정된 AODV 방식 비해서 10% 정도 성능이 향상되었다. 차후 연구에서는 멀티미디어 통신을 지원하기 위한 QoS 문제를 다루고자 한다.

참 고 문 헌

〔1〕 J. Broch, D. A. Maltz and D. B. Johnson, “Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks,” in *Proc. Int’l. Symp. Parallel Architecture, Algorithms, and Networks*, Perth, Australia, pp. 370-375, June 1999.

〔2〕 D. B. John and D. A. Malz, “Dynamic source routing in ad hoc wireless networks,” in the book *Mobile Computing*, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

〔3〕 D. Jonsson, F. Alriksson, T. Larsson, P. Johansson and J. G. Maguire, “MIPMANET - Mobile IP for Mobile Ad Hoc Networks”. *Proceedings of IEEE/ACM Workshop on Mobile and Ad Hoc Networking and Computing* (MobiHoc’00), Boston, MA, USA, pp. 75 – 85, August 2000

〔4〕 C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, Feb. 1999.

〔5〕 Y. Sun, E. M. Belding-Royer and C. E. Perkins, “Internet connectivity for ad hoc mobile networks”. *International Journal of Wireless Information Networks special issue on Mobile Ad Hoc Networks (MANETs): Standards, Research, and Applications*, 9(2), pp. 75-88, April 2002.

〔6〕 R. Prashant and K. Robin, “A hybrid approach to Internet connectivity for mobile ad hoc networks“. *Proceeding of IEEE WCNC 2003*, pp. 16 – 20: New Orleans, USA. IEEE Computer Society Press. pp. 1522 – 1527, 2003

〔7〕 C. E. Perkins and P. Bhagwat, “Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers,“ *ACM SIGCOMM: Computer Communications Review*, 24(4), pp. 234-244, Oct. 1994.

〔8〕 H. Ammari and H. El-Rewini, “Integration of mobile ad hoc networks and the Internet using mobile gateways“, *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, pp. 218-225, 2004.

〔9〕 J. Jubin and J. D. Tornow, “The DARPA packet radio network protocols,“ *Proceedings of the IEEE*, 75(1), pp. 21-32, Jan. 1987.

오 훈 (Hoon Oh) 정회원

1981년 2월 성균관대학교 전자공학과(공학사)
1992년 2월 텍사스A&M대학교 전산학과(석사)
1995년 2월 텍사스A&M대학교 전산학과(박사)
1996년 삼성전자 중앙연구소 수석연구원
2005년~현재 울산대학교 컴퓨터정보통신공학부 조교수
<관심분야> 실시간컴퓨팅, 무선통신프로토콜, 임베디드시스템, 상황인식컴퓨팅

판 안 탄 (Tan Anh Phan) 준회원

2004년 2월 하노이 공과대학교 정보공학과(졸업)
2006년 2월 울산대학교 컴퓨터공학(석사)
2007년 1월~현재 VASC Software and Media
<관심분야> 무선통신프로토콜, 임베디드시스템