위임 기능을 이용한 새로운 K-hop 클러스터 기반 Ad hoc 라우팅 구조

A Novel K-hop Cluster-based Ad hoc Routing Scheme with Delegation Functions

김 태 연* 왕 기 철**
Tae-yeon Kim Ki-cheoul Wang

요 약

기존의 ad hoc 네트워크 라우팅 프로토콜들은 노드의 동적인 특성으로 인하여 확장성에 제약을 받는다. 클러스터 기반 라우팅 프로토콜은 노드들을 클러스터로 분할하고 그들간에 계층적인 라우팅을 수행한다. 이러한 계층적인 특징은 ad hoc 네트워크 라우팅 프로토콜의 확장성을 보장한다. 그러나 기존의 k-hop 클러스터 기반 라우팅 프로토콜은 헤더의 제어 부하 등과 같은 문제점을 갖는 것으로 알려져 있다.

본 논문에서는 ad hoc 네트워크에 대해 위임 기능을 이용한 새로운 k-hop(k>3) 클러스터 기반 라우팅 구조를 제안한다. 이 구조는 멤버들을 효율적으로 관리하기 위해 트리 토폴로지를 사용한다. 헤더는 모든 멤버들에 대한 라우팅 테이블을 관리하지 않고, 멤버들 중에서 자신과 이웃하는 노드들에 대한 라우팅 테이블과 나머지 노드들에 대한 멤버 리스트만을 관리하고, 더 낮은 레벨에 있는 노드들은 중간 노드가 관리한다. 제안된 메커니즘은 클러스터 헤더의 제어 부하를 다소 줄일 수 있다.

Abstract

The existing ad hoc network protocols suffer the scalability problem due to the inherent characteristics of node mobility. Cluster-based routing protocols divide the member nodes into a set of clusters and perform a hierarchical routing between these clusters. This hierarchical feature help to improve the scalability of ad hoc network routing. However, previous k-hop cluster-based routing protocols face another problems, that is, control overhead of the cluster headers.

This paper proposes a novel k-hop cluster-based routing scheme with delegation functions for mobile ad hoc networks. The scheme employs is based on tree topology to manage cluster members in effectively. The cluster headers do not manage the routing table for whole members, while the header keeps the routing table for its neighbor members and the member list for one hop over nodes within k-hop cluster. Then the in-between leveled nodes manage the nested nodes which is structured in the lower level. Therefore, the proposed mechanism can reduce some control overhead of the cluster leaders.

Keyword : ad hoc network, cluster-based routing protocols, delegation functions, scalability

1. 서 론

사회환경이 변함에 따라 네트워크의 형태도 접근점(access points)이나 라우터와 같은 고정된 기반구조를 사용하는 구조에서 점차 ad hoc 네트워크 환경으로 바뀌어 가는 추세이다. 이동 ad hoc 네트워크는 고정된 스테이션을 고려하지 않기 때문에 서비스의 이동성과 융통성은 보장되지만 견고한 경로 설정과 신뢰된(reliable) 데이터 전송에는 취약점을 가지고 있다. 따라서 ad hoc 네트워크에 대한 대부분의 연구는 라우팅 기술에 집중되어 왔으며 지금까지 제안된 라우팅 프로토콜들은 크게 세 가지로 분류할 수 있다. on-demand 라우팅 프로토콜과 proactive 라우팅 프로토콜은 on-demand 방식과 proactive 방식을 혼용한 혼

* 정 회 원 : 서남대학교 컴퓨터정보통신학과 조교수
tykim@seonam.ac.kr(제 1저자)
** 정 회 원 : 전북대학교 대학원 컴퓨터통계정보학과 박사과정
gcwang@dcs.chonbuk.ac.kr(공동저자)

합(hybrid) 프로토콜이 있다[1-4].

on-demand나 proactive 라우팅 프로토콜은 그 구조적인 특성으로 인해 대규모 네트워크에 적용하기에는 적합하지 않는 구조이다. 다시 말해서 전자는 새로운 경로를 탐색하는 과정에서 많은 트래픽 지연이 발생하고, 후자는 정기적으로 많은 패킷이 전송됨에 따라 링크 자원을 효율적으로 사용할 수 없기 때문이다.

혼합(hybrid) 프로토콜은 모든 노드들을 존(zone)/클러스터로 분할하고, 각 존/클러스터 내에서는 proactive 라우팅 프로토콜을 사용하고 존/클러스터들간에는 on-demand 라우팅 프로토콜을 사용한다. 따라서 전술한 두 프로토콜에 비해 경로 탐색을 더 효율적으로 수행할 수 있을 뿐만 아니라 비교적 네트워크의 확장성이 보장되는 방식이다. 그러나 ZRP 프로토콜은 노드간에 과도한 제어 트래픽의 교환으로 인해 네트워크의 링크 자원을 낭비하는 단점을 가지고 있고, CBRP 프로토콜은 클러스터 멤버를 관리하고 라우팅을 수행하는 헤더(header)에 대해 과중한 부하를 주는 문제점이 있다[3, 5].

현존하는 대부분의 클러스터 기반 프로토콜은 헤더와 그의 멤버간에 직접 연결된 구조를 가정하고 있다. 그러나 클러스터의 구조가 k-hop 클러스터(k>3)인 경우에 헤더와 직접 연결되어 있지 않은 노드들에 의해 전송된 패킷이 헤더에게 도달되도록 하기 위해서는 중간 노드들의 도움이 필요하다. 따라서 헤더와 2-hop 이상 떨어진 노드가 전송한 패킷을 중계하는 중간 노드가 많이 존재하는 경우에 중간 노드들이 같은 패킷을 중복해서 전송하게 된다. 이러한 환경에서 야기될 수 있는 몇 가지 문제점을 요약해 보면 다음과 같다. 첫째, 특정 노드가 보낸 패킷을 수신한 모든 중간 노드들이 재방송하게 되면 트래픽 충돌로 인해 네트워크의 성능이 저하된다. 둘째, 헤더는 모든 멤버들에 대한 내부 경로 정보를 저장하고 전송해야할 뿐만 아니라 경로의 유지보수를 해야 하기 때문에 과중한 부하를 받게 된다. 특히 노드가 빈번하게 이동하는 환경에서 헤더가 내부 경로 정보를 갱신해야 하는 부하는 무시할 수 없다. 셋째, 헤더의 장애(이동, 전력 고갈, 방해 등)로 인해 클러스터가 파손된 경우에 이를 재구성하는데 많은 시간이 소요된다[6].

전술한 바와 같이 기존의 k-hop 클러스터는 모든 멤버들에 대해 flat한 구조를 가지고 있으며, 모든 멤버들에 대한 라우팅 테이블을 관리하고 제어 패킷을 처리하는 헤더에 대한 과부하는 네트워크의 성능을 저하시킬 뿐만 아니라 네트워크를 확장하는데 적잖은 영향을 미치게 된다. 하지만 제안된 구조는 클러스터 내의 멤버들을 트리 토폴로지를 통해 관리하기 때문에 헤더의 부하는 다소 줄일 수 있다.

본 논문은 서론에 이어 2장에서 관련연구를 기술하고, 3장에서 클러스터 기반 라우팅 구조를 설명한다. 4장에서 라우팅 프로토콜을 기술하고, 5장에서는 헤더의 부하와 방송되는 패킷 측면에서 제안된 프로토콜과 기존의 프로토콜을 비교분석한다. 마지막으로 결론과 향후 연구방향을 제시한다.

2. 관련 연구

현재까지 ad hoc 네트워크에 대해 효율적으로 경로를 탐색하고 유지보수를 수행하는 다양한 프로토콜들이 제안되었다.

Basagni[3]는 네트워크 노드의 이동 속도를 고려하여 헤더를 결정하는 두 가지 알고리즘(DCA (Distributed Clustering Algorithm), DMAC (Distributed and Mobility-Adaptive Clustering algorithm))을 제안하였다. 이는 모든 노드들에게 서로 다른 가중치를 할당하고, 연결된 노드들의 형태에 따라 해당 클러스터의 헤더를 결정하는 방식을 기술하였다.

ZRP[4]는 네트워크 내의 모든 노드들이 HELLO 메시지를 방송하여 존을 구성하는 방식을 제안하였다. 경로 탐색 시에 존간에 겹쳐지는 모든 노드

들이 경로 요청 패킷을 재방송하기 때문에 많은 패킷이 이동하게 된다. 이러한 문제를 줄이기 위해 Jae-Pil[7] 등은 존이 겹쳐지는 부분을 줄이기 위해 경계선 개념을 사용하여 패킷을 방송하는 메커니즘을 제안하였지만 교환되는 패킷의 량을 무시할 수 없는 구조이다. 반면에 CBRP[3]는 클러스터 헤더만이 경로 요청 패킷을 방송할 수 있도록 하는 방식을 제안하였지만 클러스터 헤더는 모든 멤버에 대한 라우팅 정보를 저장하고 전송뿐만 아니라 경로의 유지보수를 수행하는 역할을 하기 때문에 많은 부하를 받게 된다. 결국 존 구조에서의 과도한 패킷의 이동과 클러스터 구조에서의 헤더의 과부하는 네트워크를 확장하는데 방해 요인이 된다.

Elizabeth[8]은 ad hoc 네트워크에서 클러스터를 기반으로 하는 계층적인 라우팅 기술을 제안한 혼합 프로토콜로서 클러스터간에 다중 게이트웨이를 두어 견고한 경로가 보장되고, 경로 파손을 국부적으로 복구할 수 있는 프로토콜을 제안하였다. 그러나 이 구조는 견고한 경로를 보장하고 장애 발생시에 효율적인 복구하는 알고리즘만을 제시했을 뿐 확장성을 지원하는 메커니즘을 기술하지 않았다.

Fei LI[9] 등은 이동 ad hoc 네트워크에서 클러스터 스핀 토폴로지를 이용한 이동 멀티 캐스트 에이전트 방식을 제안하였다. 이곳은 각 클러스터는 서로 겹쳐지지 않는다는 가정 하에서 클러스터내의 멤버를 관리하는 다중의 에이전트를 두는 방식으로 헤더의 과부하를 줄일 뿐만 아니라 신뢰성 있는 제어 정보를 전송하고 관리하는 라우팅 프로토콜이다. 이 구조는 헤더의 부하를 줄이기 위한 메커니즘만을 언급하였으며 확장성에 관한 알고리즘은 제시하지 않았다.

Dongkyun, kim[5] 등은 k-hop 클러스터의 구조에 대한 일반화된 라우팅 프로토콜을 제시하였다. 그러나 k 값이 커짐에 따라 헤더는 모든 이동 멤버들에 대한 라우팅 테이블을 관리해야 하는 부하가 가중되는 문제점을 가지고 있다.

3. 클러스터 기반 라우팅 구조

3.1 용어 정의

클러스터 기반 라우팅 구조는 모든 이동 노드들을 클러스터라고 하는 도메인들로 분할하고, 각 도메인은 헤더에 의해 관리된다. 클러스터 멤버인 모든 노드는 인접 노드에 대한 정보를 정기적으로 방송하여 헤더로 하여금 내부 라우팅 정보를 수집할 수 있도록 할뿐만 아니라 실제 데이터를 송수신하고, 인접 노드가 라우팅 규칙을 위반하는지를 감시하고 통보하는 기본적인 역할을 한다. 이러한 구조에서는 클러스터의 모든 멤버들에 대한 라우팅 테이블을 관리하는 헤더는 과중한 부하를 받기 때문에 제안된 구조에서는 멤버들을 계층적으로 관리하는 트리 토폴로지를 사용한다. 클러스터를 구성하는 멤버들은 자신의 역할에 따라 단 노드 $T_{i,j}(i, j>=0)$와 멤버를 관리하는 헤더 노드 $L_{0,j}(j>=0)$, 단 노드 $T_{i,j}$와 노드 $L_{0,j}$의 사이에 존재하는 중간 노드 $L_{i,j}(i>0, j>=0)$, 클러스터간을 연결하는 게이트웨이 $G_{i,j}(i, j>=0)$로 분류되는데 각 노드에 대한 정의는 다음과 같다.

단 노드 $T_{i,j}$. 클러스터의 멤버로서 기본적인 역할만을 수행하는 노드이다.

헤더 $L_{0,j}$. 클러스터의 모든 멤버(단 노드, 중간 노드 $L_{i,j}$, 게이트웨이 노드 $G_{i,j}$, 인접 클러스터의 헤더)들의 ID 리스트들을 관리하고 제어 패킷(라우팅, 에러 처리 등)을 처리하는 역할을 담당하는 클러스터 헤더이다. 그러나 헤더는 멤버 중에서 1-hop에 있는 노드들에 대해서만 라우팅 테이블은 유지하고, 2-hop 이상 떨어진 노드들에 대해서는 멤버들의 리스트만을 관리하기 때문에 멤버들에게 메시지를 전달할 때는 방송 기능을 사용한다.

중간 노드 $L_{i,j}$. 노드 $L_{0,j}$ 권한의 일부분을 위임받아 수행하는 클러스터의 멤버로서 자신과 직접 통신할 수 있는 노드 중에서 상위 노드 $L_{i-1,j}$가

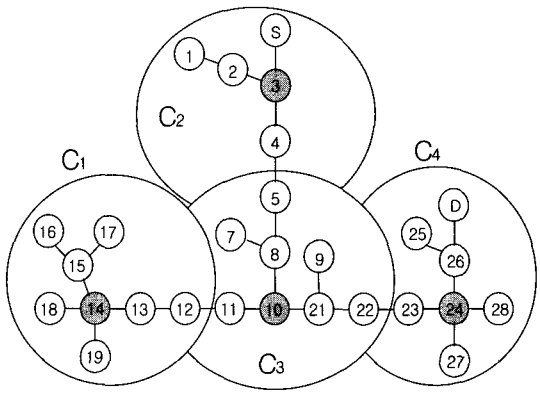
직접 관리할 수 없는 노드와 하위 노드나 노드 $L_{i+1,j}$가 관리는 노드들의 리스트를 관리하는 중간 노드이다. 또한 노드의 이동에 대한 정보를 상위 노드에게 통보하고, 상위 노드로부터 받은 메시지를 자신이 관리하는 노드들에게 전달할 때는 하위 중간 노드가 관리하고 있는 노드들에 대한 라우팅 테이블은 가지고 있기 않기 때문에 방송 기능을 사용한다.

게이트웨이 노드 $G_{i,j}$. 게이트웨이는 두 개 이상의 클러스터에 의해서 겹쳐진 지역에 있는 이동 노드로서 클러스터간에 패킷을 중계하는 역할을 담당하는 노드이다.

failed 노드. ad hoc 네트워크를 구성하는 노드이지만 어떤 클러스터의 멤버에도 포함되지 않은 노드이다.

그림 1과 같이 클러스터를 구성하는 멤버들의 연결 형태에 따라 각 노드는 단 노드(S, D, 1, 7, 9, 16, 17, 18, 19, 25, 27, 28)와 헤더(3, 10, 14), 중간 노드(2, 4, 8, 11, 13, 15, 23, 26), 게이트웨이 노드(5, 12, 22)로 구분된다.

클러스터의 모든 멤버는 주기적으로 인접 노드들과 자신의 존재를 알리는 ID를 교환하고, 자신의 역할과 상위 노드인 $L_{i,j}$들의 리스트 PL(Parent List)를 관리한다. 노드 $L_{0,j}$는 모든 멤버들에 대한 멤버 리스트 ML(Member List)를 관리하는데 자신과 1-hop으로 연결된 노드들에 대해서는 라우팅 정보를 유지한다. 또한 멤버 리스트 중에서 직접 관리하지 않는 노드들을 관리하는 중간 노드 $L_{1,j}$들의 리스트 AL(middle Agent List)과 인접 클러스터를 서로 연결해주는 게이트웨이 $G_{i,j}$들의 리스트 GL(Gateway List), 게이트웨이에 연결된 인접 클러스터의 헤더 $L_{0,j+1}(j>0)$의 리스트 LL(Leader List)을 관리한다. 중간 노드 $L_{i,j}$는 자신이 관리하는 하위 멤버들의 리스트 CL(Child List)와 상위 노드 $L_{i-1,j}$들의 리스트 PM를 관리하고, 게이트웨이 노드 $G_{i,j}$는 자신을 게이트웨이 역할로 지정한 노드 $L_{0,j}$들의 리스트 LL와 PL을 관리한다. 예를 들어, 그림 1의 클러스터 C_1에서 노드 17은 PL={15}, 노드 14는 ML={12, 13, ..., 19}와 AL={13, 15}, GL={12}, LL={10}, 노드 15는 PL={14}와 ML={16, 17}, 노드 12는 PL={11, 13}과 LL={10, 14}를 관리한다.



〈그림 1〉 클러스터 구성 예

클러스터 내에 동일한 서브그룹 멤버들을 관리할 수 있는 노드 $L_{i,j}$가 여러 개 있는 경우에 노드 $L_{i-1,j}$는 효율적인 운영을 위해 그 수를 제한할 수 있다. 게이트웨이 역할을 하는 노드가 여러 개 있는 경우에도 노드 $L_{0,j}$은 수를 제한할 수 있고, 실제로 패킷을 중계할 수 있는 게이트웨이를 선택하는 메커니즘은 노드 $L_{i,j}$를 선택하는 것과 같이 운영된다[8].

3.2 초기 클러스터 구성

ad hoc 네트워크의 모든 노드들은 클러스터를 구성하기 위해 인접 노드들에게 HELLO 메시지를 방송한다. 클러스터 구성 규칙에 의해서 멤버 중에서 특정 노드가 나머지 멤버들을 관리하는 헤더가 된다.

편의상 클러스터를 구성하는 형태는 노드 $L_{0,j}$ (j>=0)와 그 멤버간에 최대 2-hop으로 연결된 형태인 4-hop 클러스터를 가정한다. 또한 동일한 서브그룹 멤버들을 관리하는 중간 노드 $L_{i,j}$나 인

접 클러스터의 노드 $L_{0,j+1}$에게 패킷을 중계하는 게이트웨이 노드 $G_{i,j}$가 다른 지역으로 이동하거나 장애를 받은 경우에 최소한의 경로 지속성을 보장하기 위해 그 수를 각각 최대 두 개로 한정한다. 초기 4-hop 클러스터를 구성하는 알고리즘은 다음과 같다.

1. 초기 상태로 네트워크의 모든 노드에 failed 역할이 주어진다.
2. 각 노드는 정기적으로 인접 노드에게 HELLO 메시지를 방송한다.
3. 메시지를 받은 인접 노드들 중에서 가장 낮은 ID를 가진 노드를 $L_{0,j}$로, 나머지 노드는 단 노드 $T_{i,j}$로 지정한다.
4. 노드 $L_{0,j}$는 멤버 중에서 자신과 2-hop 떨어진 노드를 연결해 주는 노드를 노드 $L_{1,j}$로, 인접 클러스터를 연결해 주는 노드를 게이트웨이 노드 $G_{i,j}$로 지정한다. 단, 동일한 서브그룹 멤버들을 관리하는 노드와 인접 클러스터의 멤버에게 패킷을 중계하는 노드의 수가 2개를 넘은 경우에는 해당 노드를 $T_{i,j}$로 지정한다.
   (1) 특정 클러스터가 다른 클러스터의 범위 내에 포함되는 경우에 하나의 클러스터로 통합하고, 포함된 클러스터의 노드 $L_{0,j}$는 노드 $L_{1,j+1}$로 지정한다.
   (2) 두 인접 클러스터 노드 $L_{0,j}$의 ID에 따라 단 노드의 역할이 갱신될 수 있다. 예를 들어, 자신과 직접 연결된 단 노드가 인접 클러스터의 멤버와 직접 연결되어 있고, 상대편 노드 $L_{0,j}$의 ID가 자신보다 높은 경우에 단 노드를 노드 $L_{1,j}$로 지정하고, 인접 노드 $L_{0,j}$를 게이트웨이로 지정한다. 여기에서, 게이트웨이로 지정된 노드는 이전 멤버들에게 갱신된 자신의 역할을 통보한다.
   (3) 임의의 노드가 클러스터의 노드 $L_{0,j}$와 2-hop으로 연결되어 있으면서 인접 클러스터의 노드 $L_{0,j+1}$과 2-hop으로 연결된다면 해당 노드는 게이트웨이의 역할을 한다.
5. 클러스터가 완성될 때까지 단계 4를 반복한다.

4. 개선된 라우팅 프로토콜

4.1 경로 탐색

본 논문에서 제안된 경로 탐색 프로토콜은 기존의 메커니즘과 유사하게 클러스터 내부에서는 내부 라우팅 경로를 설정하는 proactive를 사용하고 클러스터간에는 외부 라우팅 경로를 설정하는 on-demand 알고리즘을 사용한다. 송신자 노드로부터 목적지 노드까지의 경로를 탐색할 때 노드 $L_{0,j}$는 클러스터 내의 멤버들에 대한 멤버 리스트를 가지고 있기 때문에 외부 경로 탐색을 수행할 것인지를 판단할 수 있다.

예를 들어, 송신자 노드가 특정 목적지 노드에게 데이터를 전송하고자 하는 경우에 자신을 관리하는 노드 $L_{0,j}$에게 경로 요청 패킷을 보낸다. 패킷을 수신한 노드 $L_{0,j}$는 목적지 노드가 자신의 클러스터 내에 존재하는지를 멤버 리스트 ML을 검사한다. 만일 목적지가 존재하는 경우에는 라우팅 경로를 탐색하기 위해 내부 멤버에게 패킷을 방송하여 목적지 노드에게 전달한다. 만일 목적지 노드가 자신의 멤버가 아닌 경우에는 목적지 노드까지의 경로를 찾기 위해 게이트웨이들에게 자신의 ID가 첨가된 패킷을 border-casting하여 경로 탐색을 계속한다.

예를 들어, 그림 1과 같이 송신자 노드 S가 목적지 노드 D에 데이터 패킷을 보내기 위해서는 목적지로의 경로를 알고 있어야 한다. 따라서 원하는 경로가 자신의 경로 캐시에 저장되어 있지 않는 경우에 노드 S는 경로를 찾기 위해 경로 탐색 알고리즘을 수행한다. 즉, 노드 S는 노드 D에

게 데이터를 전송할 수 있는 경로를 얻기 위해 경로 요청 패킷을 보냄으로서 경로 탐색이 시작된다. 만일 S가 단 노드이면 상위 노드에게 경로 요청 패킷을 전송한다. 패킷을 수신한 노드 $L_{1,j}$나 $L_{0,j}$는 다른 노드에 의해서 최근에 수신한 패킷인지를 검사한다. 해당 패킷이 최근에 받았던 것이라면 폐기하고, 그렇지 않은 경우에 목적지 노드가 자신의 클러스터 내의 멤버인지를 검사한다. 멤버가 아닌 경우에는 수신한 패킷의 헤더 필드에 자신의 ID를 붙인 다음 패킷을 게이트웨이 노드 $G_{i,j}$들에게 방송한다. 송신자 S가 목적지 D까지의 경로를 탐색하는 절차는 다음과 같다.

1. 노드 S는 노드 D를 지정한 경로 요청 패킷을 상위 노드에게 전송한다.
2. 하위 노드로부터 요청 패킷을 수신한 상위 노드는 목적지 노드가 자신의 멤버인지를 검사한다.

(1) 상위 노드가 $L_{0,j}$인 경우

(a) 목적지 노드가 자신이 관리하는 멤버인 경우 패킷을 방송한다.

- 목적지 노드가 1-hop으로 연결된 경우, 노드 $L_{0,j}$는 경우 최적의 라우팅 경로를 응답(reply) 패킷에 첨가한 다음 수신한 반대 방향으로 보낸다.
- 목적지 노드가 1-hop으로 연결되어 있지 않는 경우, 노드 $L_{0,j}$는 하위 노드 $L_{1,j}$들에게 패킷을 방송하여 목적지에 대한 경로를 얻는 다음에 최적의 라우팅 경로를 응답 패킷에 첨가한 다음 수신한 반대 방향으로 보낸다.

예를 들어, 그림 1에서 C4내의 헤더인 24는 목적지 노드가 28인 경우 목적지에 대한 경로를 알고 있기 때문에 바로 28에게 보낸다. 그러나 실제 예와 같이 노드 D가 목적지인 경우에는 멤버라는 것은 알 수 있지만 D에 대한 정확한 경로를 모르기 때문에 노드 26과 27, 28에게 요청 패킷을 방송하여 경로를 얻는다.

(b) 자신의 멤버가 아닌 경우 노드 $L_{0,j}$는 헤더는 수신한 패킷 헤더에 자신의 ID를 첨가한 후에 게이트웨이 노드 $G_{i,j}$에게 방송한다. C_3내의 노드 10은 D가 자신의 멤버가 아니기 때문에 게이트웨이 노드 12와 22에게 방송한다.

(2) 상위 노드가 $L_{1,j}$인 경우

목적지 노드가 자신의 멤버가 아닌 경우는 패킷을 노드 $L_{0,j}$에게 그대로 중계하고, 멤버인 경우에는 패킷 헤더에 목적지 노드까지의 최적의 내부 경로를 헤더에 첨가한 다음 요청 패킷을 수신한 반대 방향으로 전송한다. 예를 들어, 그림 1의 클러스터 C_1내의 노드 13과 C_3내의 노드 8은 D에 관한 정보를 가지고 있지 않기 때문에 각각 상위 노드 14와 10에게 전달한다.

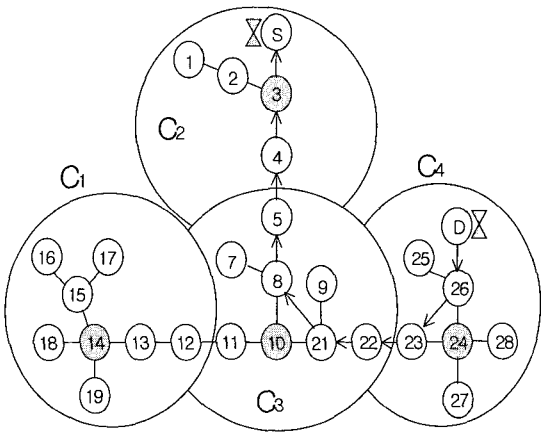
3. 상위 노드로부터 요청 패킷을 수신한 노드 $L_{1,j}$는 목적지 노드가 자신의 멤버가 아닌 경우는 패킷을 폐기하고, 멤버인 경우에는 패킷 헤더에 목적지까지의 경로를 헤더에 첨가한 다음 요청 패킷을 상위 노드에게 전송한다. 예를 들어, 그림 1에서 클러스터 C_4내의 노드 27과 28은 D가 자신의 하위 노드에 있지 않기 때문에 요청 패킷을 폐기 하지만 노드 26은 노드 D에게 요청 패킷을 전달하고 D로부터 받은 응답 패킷을 상위 노드 24에게 보낸다.
4. 게이트웨이 노드 $G_{i,j}$는 수신한 패킷을 자신과 연결된 패킷을 전송한 헤더 노드를 제외한 모든 인접 클러스터의 헤더인 노드 $L_{0,j+1}$에게 재방송한다. 그림 2의 클러스터 C_3의 노드 12는 14에게, 노드 22는 24에게 요청 패킷을 보

낸다.

5. 목적지 노드를 찾을 때까지 단계 2와 3, 4를 반복한다.

4.2 경로 설정

경로 설정은 송신자가 목적지 노드로 데이터를 전송할 수 있는 실제 경로를 지정하는 절차이다. 목적지 노드를 멤버로 관리하는 노드 $L_{0,j}$나 $L_{1,j}$는 경로 요청 패킷을 수신하면 목적지 노드에게 전송하고, 경로 응답 패킷의 헤더에 목적지 노드에 대한 최적의 내부 라우팅 경로를 경로 리스트 필드에 기술한 다음 수신한 반대 방향으로 유니캐스트한다. 경로 요청 패킷을 중계하는 다른 클러스터의 노드 $L_{0,j}$는 자신의 클러스터내의 최적 내부 라우팅 경로를 수신한 응답 패킷 헤더의 필드인 경로 리스트의 내용 앞부분에 붙인 후에 다시 노드 S 방향으로 전송한다.



〈그림 2〉 경로 설정

목적지 노드 D로부터 노드 S까지의 응답 패킷이 전달되는 도중에 경로 설정을 하는 과정은 다음과 같다.

1. 목적지가 아닌 모든 단 노드는 수신한 경로 요청 패킷을 폐기한다.
2. 목적지 노드를 관리하는 헤더나 중간 노드는 목적지 노드에 경로 요청 패킷을 전송하고 응답 패킷 헤더의 경로 리스트 필드에 최적의 내부 라우팅 경로를 기술한 후에 요청 패킷을 전송한 게이트웨이나 송신자 노드 방향으로 보낸다. 그림 2에서 노드 10과 24는 각각 노드 8과 23에게 응답 패킷을 전송한다.
3. 송신자 노드 S가 아닌 게이트웨이 노드가 응답 패킷을 받으면 응답 패킷을 인접 클러스터의 노드 $L_{0,j-1}$에게 보낸다. 노드 $L_{0,j-1}$은 수신한 응답 패킷의 필드에 최적의 내부 라우팅 경로를 첨가한 다음에 다시 요청 패킷을 수신한 반대 방향으로 전송한다. 노드 24와 10은 내부 경로가 각각 노드 26에서 23으로, 노드 21에서 8로 기술된 정보를 응답 패킷에 첨가하여 전송한다.
4. 요청 패킷이 송신자 노드 S에 도달할 때까지 3번 과정을 반복한다.

4.3 경로 유지보수

경로 유지보수는 클러스터 내의 특정 멤버가 이동함에 따라 발생되는 노드들에 대한 토폴로지의 변화에 대해 대처해야할 뿐만 아니라 전송 장애로 인해 예정된 경로가 파손되어 데이터 패킷을 원하는 목적지에 전송할 수 없는 문제를 효율적으로 복구하는 메커니즘이다.

클러스터를 유지하는데 있어서 특정 노드가 다른 곳으로 이동하는 경우에 토폴로지가 변경된 경우이므로 헤더는 관련된 라우팅 정보를 갱신해야 한다. 따라서 노드가 클러스터 내의 서브그룹에서 다른 서브그룹으로 이동하는 경우에도 헤더는 라우팅 테이블을 갱신해야 한다. 이러한 처리는 노드가 빈번하게 이동하는 네트워크 환경에서는 헤더의 부하를 가중시키는 결과를 초래한다. 하지만 제안된 방식에서는 노드가 서브그룹간의 이동이 있는 경우에 헤더에게는 영향을 미치지

않고 국부적으로 해결할 수 있는 메커니즘을 지원한다. 다시 말해서 인접 서브그룹으로의 이동이 있는 경우에 헤더가 관리하는 멤버 리스트를 갱신하지 않고 인접 중간 노드들($L_{i,j}$와 $L_{i,j+1}$)의 멤버 리스트만을 갱신하면 되기 때문에 헤더의 부하를 줄일 수 있게 된다.

또한 대부분의 라우팅 경로의 파손은 대부분 전체 경로의 국부적인 장애 즉, 해당 클러스터 멤버인 노드의 탈퇴(leave)나 전력 고갈 등으로 인해 파괴된다. 제안된 라우팅 구조는 전송 장애가 발생한 경우에 파손 경로를 국부적으로 복구할 수 있는 메커니즘을 지원한다. 정보 패킷을 보낼 때 송신자는 패킷의 헤더에 전송될 경로를 기술한 다음에 보낸다. 따라서 중간 노드는 헤더에 지정된 다음 노드에 데이터를 전송하고 해당 노드가 성공적으로 수신했는지를 확인하는 절차를 거친다. 수신 확인 응답을 제대로 수신하지 못하면 자신과 다음 노드간의 링크가 파손된 것으로 간주하고 노드 $L_{0,j}$에게 링크 장애(link failure) 패킷을 전달한다. 링크 장애 패킷을 수신한 노드 $L_{0,j}$는 경로 캐쉬에 저장된 사용중인 경로 정보를 제거하고, 다른 최적의 경로를 선택하여 해당 노드에게 전달한다. 노드 $L_{0,j}$가 다른 최적의 경로를 찾을 수 없는 경우에는 송신자 노드 S에게 경로 에러 패킷을 전달하여 새로운 경로를 탐색할 수 있도록 한다.

경로 유지보수 처리는 경로상의 지역 클러스터 단위로 손상된 경로를 자치적으로 복구하는 메커니즘을 지원하는데 그 복구 절차는 다음과 같다.

1. 라우팅 경로에 의해 데이터 패킷을 중계하는 멤버 중에서 장애가 발생한 멤버의 바로 앞 노드는 노드 $L_{0,j}$에게 링크 장애 패킷을 보낸다.
2. 링크 장애 패킷을 받은 노드 $L_{0,j}$는 국부적으로 장애를 해결할 수 있는 새로운 경로가 존재하는지를 검사한다. 경로가 존재하면 아래 (1)과 (2), (3) 중에서 하나를 수행하여 정보 패킷을 전송할 수 있도록 한다. 그렇지 않으면 송신자 S에게 경로 에러 패킷을 보내 다시 경로 탐색을 수행할 수 있도록 한다.
   (1) 노드 $L_{0,j}$와 1-hop으로 연결된 노드에 장애가 발생한 경우는 노드 $L_{0,j}$는 새로운 라우팅 경로를 정보 패킷을 전송한 게이트웨이 노드 $G_{i,j-1}$에게 전송한다.
   (2) 헤더와 2-hop으로 연결된 노드에 장애가 발생한 경우는 하위 노드 $L_{i,j}$에게 링크 장애 패킷을 보내 서브그룹 멤버 리스트를 받아서 새로운 라우팅 경로를 생성한 다음 정보 패킷을 전송한 게이트웨이 노드 $G_{i,j-1}$에게 전송한다.
   (3) 정보 패킷을 중계하는 게이트웨이에 장애가 발생했지만 대행할 수 있는 다른 게이트웨이가 존재한 경우에 노드 $L_{0,j}$는 해당 게이트웨이 노드 $G_{i,j+1}$에게 변경된 라우팅 경로를 보낸다.
3. 데이터를 중계하는 노드 $L_{0,j}$에 장애가 발생한 경우에 장애가 발생한 멤버의 바로 앞 노드는 송신자에게 경로 에러 패킷을 보내 다시 경로 탐색을 수행할 수 있도록 한다.

5. 성능 분석

이 장에서는 Jang M. 등이 제안한 CBRP[3]과 기존의 flat한 k-hop(k>3) 클러스터, 제안된 클러스터 구조에 대해 멤버들을 관리하고 제어 패킷을 처리하는 헤더의 부하와 네트워크의 성능을 저하시키는 트래픽의 량 측면에서 서로 비교 분석한다.

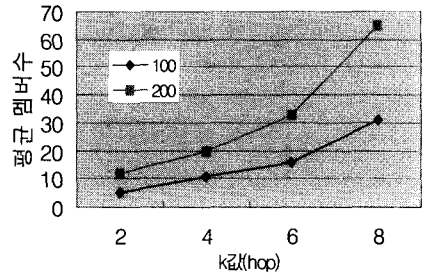
k-hop 클러스터 구조에 대한 시뮬레이션은 다음과 같은 가정하에서 10초 동안 수행된다. 첫째, 이동 노드들(100 또는 200개)은 정해진 구역(100m×100m)내에 불규칙하게 설치되어 있다. 둘째, 노드의 패킷 전송 범위는 10m이고, 특정 노드들(10 또는 30개)은 동시에 임의의 방향으로 고정된 속도(0 또는 5㎧)로 이동한다. 셋째, 주어

진 시간 동안에 임의의 노드들(0~5개)은 동시에 경로 탐색을 요청한다. 마지막으로 헤더가 라우팅 정보를 관리하는데 필요한 기억공간에 대한 비용은 4n+4(n:멤버수) 바이트로 계산하고, 라우팅 정보를 갱신하는데 소요되는 계산 부하는 노드당 1로 계산한다.

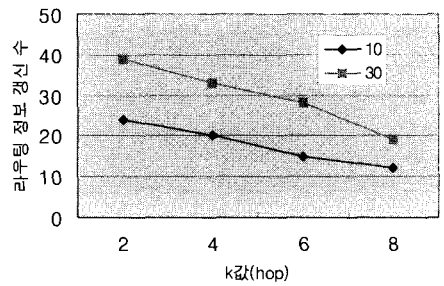
그림 4는 시뮬레이션을 수행하기 전에 구성된 클러스터의 초기 상태로서 k값에 대한 각 클러스터들의 평균 멤버수를 나타낸 것이다. 이러한 결과는 전체 노드 수에 관계없이 노드들이 일정하게 분포되어 있음을 의미한다.

그림 5는 시뮬레이션을 수행하는 동안 특정 노드들(10와 30개)이 임의의 방향으로 이동했을 때 k값과 각 클러스터에 대해 평균 라우팅 정보를 갱신하는 횟수에 대한 관계를 나타낸 것이다. 이 결과를 통해 알 수 있듯이 k값이 증가함에 따라 노드의 이동이 클러스터내에서 이루어지기 때문에 라우팅 정보를 갱신할 필요가 없어 그 횟수가 줄어든다. 즉, 같은 조건에서 k값이 큰 경우에 이동 노드가 자신의 클러스터를 벗어날 확률이 적고, 반대로 k값이 작은 경우에는 노드가 인접 클러스터로 이동할 확률이 높다는 것을 의미한다.

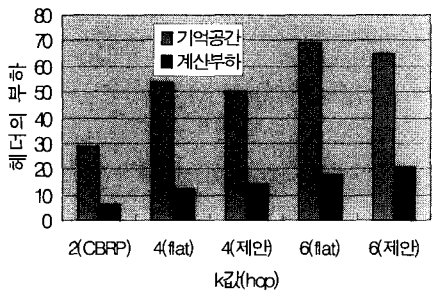
그림 6은 헤더가 자신의 멤버들에 대한 라우팅 정보를 저장하는데 필요한 기억 공간과 계산 처리로 인해 발생하는 부하를 나타낸 것이다. k값이 커지면 네트워크내의 클러스터 수는 줄어들지만 헤더가 관리하는 멤버 수는 늘어나기 때문에 기억공간의 사용 측면에서 보면 헤더의 부하는 증가한다. 그러나 제안된 구조가 기존의 flat한 구조보다 더 적은 기억공간을 필요로 하지만 계산 처리 부하는 더 높게 나타났다. 기억공간 항목의 차이는 기존의 방식이 모든 멤버들에 대한 라우팅 정보를 관리하지만, 제안된 방식은 직접 인접된 노드들에 대한 정보만을 관리하고 2-hop 이상 떨어진 멤버들에 대해서는 그 리스트만 관리하기 때문에 기억공간을 적게 차지하게 된다. 그리고



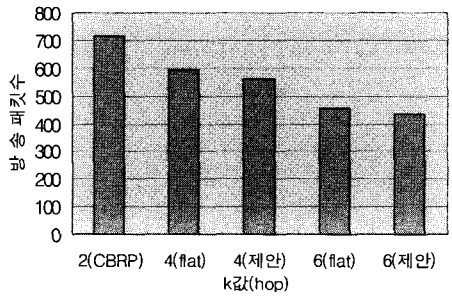
〈그림 3〉 k값에 따른 클러스터내의 평균 멤버 수



〈그림 4〉 평균 라우팅 정보 갱신 횟수



〈그림 5〉 K값에 따른 헤더의 부하



〈그림 6〉 노드간에 교환하는 패킷 수

계산 부하의 차이는 제안된 구조가 flat한 구조에서는 사용하지 않는 위임 기능을 사용하기 때문이다. 다시 말해서, 헤더의 부하에 중간 노드가 멤버들에 대한 라우팅 정보를 갱신하는데 소요되는 처리 시간을 더했기 때문이다.

그림 7은 경로 탐색 시에 k값과 전체 네트워크 내의 모든 노드간에 교환되는 패킷 수와의 관계를 나타낸 것이다. 제안된 구조는 수신한 패킷을 재방송할 수 있는 권한을 가진 노드를 헤더와 중간 노드, 게이트웨이로 제한하기 때문에 패킷을 중복 방송하는 것을 최소화할 수 있지만, 기존의 방식에서 모든 노드는 최근에 수신하지 않은 패킷은 모두 방송하기 때문에 패킷을 중복해서 전송하게 된다. 결과적으로, k값이 작아짐에 따라 교환되는 패킷 수가 늘어나 링크 채널 자원을 많이 낭비하게 되어 네트워크의 성능을 저하시키는 결과를 초래한다. 또한 노드들이 제어 패킷들을 처리하는 것보다 패킷을 전송하는데 베터리의 전력을 더 많이 소모한다는 점에서 헤더의 부하는 가중된다.

지금까지 헤더의 부하와 전송되는 트래픽의 량을 비교분석한 결과, 작은 규모의 지역에 대해 노드가 균등하게 분포되어 있는 경우에는 k값이 작은 것이 좋고, 지역이 넓은 지역에 노드들이 국부적으로 집중되어 있는 경우에는 k값이 큰 것이 효율적임을 알 수 있다. 그리고 대규모 클러스터 기반 ad hoc 네트워크에서 큰 k값을 사용하는 구조에서 발생하는 헤더의 부하 문제를 줄이기 위해 고성능의 처리기와 베터리, 대용량의 기억장치 등을 가진 노드를 헤더의 역할을 하도록 하고, 저기능의 노드들을 멤버로 사용하는 것도 좋은 방법이 될 수 있다.

6. 결 론

본 논문에서 이동 노드들을 계층적으로 관리하여 불필요한 트래픽과 헤더의 과부하를 줄임에 따라 네트워크의 확장성을 보장할 수 있는 개선된 클러스터 기반 라우팅프로토콜을 제안하였다. 또한 제안된 프로토콜은 클러스터를 효율적으로 관리할 뿐만 아니라 경로의 견고성이 보장되고, 국부적인 경로의 파손이 발생한 경우에 클러스터 내의 헤더에 의해 복구될 수 있는 특징을 가지고 있다. 세부적인 특징들을 요약하면 다음과 같다.

첫째, 클러스터 내의 단 노드가 정기적으로 방송하는 라우팅 정보의 범위를 제한한다. 즉, 새로운 노드가 클러스터의 가입을 요청하거나 기존 인접 노드가 통신 범위를 이탈, 장애로 인해 연결이 끊김 등과 같이 인접 노드에 대한 정보가 변경된 경우에만 해당 헤더나 중간 노드에게 전달한다.

둘째, 클러스터내의 멤버들간에 패킷을 전달할 때는 중복 전송되는 것을 피하기 위해 헤더와 중간 노드만이 패킷을 재방송할 수 있는 권한을 준다.

셋째, 에이전트 개념을 사용함에 따라 헤더의 장애로 인한 클러스터의 파손을 국부적으로 한정시킬 수 있다. 따라서 클러스터를 재구성하는데 걸리는 시간을 단축시킬 수 있고, 헤더의 역할이 하위 계층의 특정 노드에게 부분적으로 위임되었기 때문에 헤더의 과부하를 줄일 수 있다.

향후 연구로는 실제 환경에서 더 구체적인 성능 분석과 제안된 구조에 적합한 보안 메커니즘의 설계가 필요하다.

참 고 문 헌

[1] Sethi, P., Barua, G., "CRESQ: Providing QoS and Security in Ad Hoc Networks, " In Proceedings of the Eleventh Euromicro Conference on Parallel, Distributed and Network-based Processing(Euro-PDP'03), 2003.

[2] Ramasubramanian, V., Haas, J., Sier, E., "SHARP:A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks,"

ACM Mobile Ad hoc Networking & Computing, Jun. 2003.

[3] Jiang, M., Li, J., Tay, Y., "Cluster Based Routing Protocol(CBRP)," Internet Draft, Jul. 1999.

[4] Haas, Z. J., Pearlman, M. R., "The Zone Routing Protocol(ZRP) for Ad Hoc Networks," Internet Draft, Nov. 1997.

[5] Kim, D. K., Ha, S. J., and Choi, Y. H., "K-hop Cluster-based Dynamic Source Routing in Wireless Ad-Hoc Packet Radio Network," IEEE VTC '98, 1998.

[6] Marti, S., Giulin-Lai, T. J., Baker M., "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," MOBICOM 2000, 2000.

[7] Yoo, J. P., Kim, K. C., Han, S. Y., "Independent Zone Setup Scheme for Re-configurable Wireless Network," ICCS, Vol. 2, 2003.

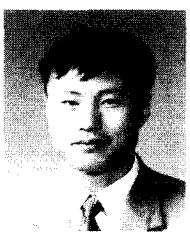
[8] Belding-Royer, E. M., "Hierarchical Routing in Ad Hoc Mobile Networks," Wireless communication & Mobile Computing, Vol. 2, No. 5, 2002.

[9] LI, F., Wang, X., Xue, X., "A Mobile Multicast Algorithm Using Agents for Mobile Ad-hoc Users," In Proceedings of the 17th International Conference on Advanced Information Networking and Applications(AINA'03), 2003.

◐ 저 자 소 개 ◑

김 태 연

1985년 전남대학교 계산통계학과 졸업(학사)
1988년 전남대학교 대학원 전산통계학과 졸업(이학석사)
1996년 전남대학교 대학원 전산통계학과 졸업(이학박사)
1996년~현재 서남대학교 컴퓨터정보통신학과 조교수
관심분야 : 네트워크 보안, 이동 컴퓨팅, 네트워크 관리
E-mail : tykim@seonam.ac.kr



왕 기 철

1997년 광주대학교 전자계산학과 졸업(학사)
2000년 목포대학교 대학원 멀티미디어 공학과 졸업(석사)
2001~현재 전북대학교 대학원 컴퓨터통계정보학과 박사과정
관심분야 : Ad-hoc 네트워크, 모바일 컴퓨팅. 무선 네트워크 보안
E-mail : gcwang@dcs.chonbuk.ac.kr