이동 Ad Hoc 네트워크에서 클러스터 기반의 DSDV 라우팅 프로토콜

정회원 오 훈*

Cluster-Based DSDV Routing Protocol in Mobile Ad Hoc Networks

Hoon Oh* *Regular Member*

요 약

이동 Ad Hoc 네트워크에서 DSDV개념을 적용하는 클러스터 기반 c-DSDV 프로토콜을 제안한다. 각 클러스터에는 멤버들을 관리하는 클러스터헤드가 존재하고 클러스터헤드들끼리 형성하는 상위 계층 네트워크 백본에 DSDV 개념을 수정 및 적용하였다. 클러스터헤드들은 갱신 메시지를 이웃 클러스터헤드들과 교환함으로써 전역 라우팅 테이블을 구성한다. 라우팅 테이블은 전체 클러스터헤드의 수에 해당되는 적은 수의 엔트리를 갖는다. 각 갱신 메시지는 최대 3홉 거리에 있는 이웃 클러스터헤드들에게 까지 플러딩되기 때문에 토폴로지 수렴 범위가 DSDV 및 CGSR의 수렴범위에 비해 최소 9배 커지고 라우팅 정보의 정확도가 크게 향상된다. 하지만, 클러스터헤드들만 갱신 메시지를 생성하기 때문에 실제 오버헤드는 DSDV의 오버헤드와 비슷하다. 패킷 전송률이 32~50% 정도 개선되었다.

Key Words : Ad hoc network, Proactive, Routing protocol, Cluster, Clusterhead.

ABSTRACT

A novel c-DSDV routing protocol is proposed in clustered mobile ad hoc networks. Clusterheads that manage members in their own cluster construct a higher layer backbone to which the DSDV concept is applied. Each clusterhead maintains its own global routing table by exchanging Update Request (UREQ) messages with its neighboring clusterheads. A number of entries in the table is as small as a number of clusterheads unlike a number of nodes in DSDV. Since a UREQ message travels from one clusterhead to all its neighboring clusterheads that are at most 3 hops away, the topology convergence range by each UREQ message is at least 9 times as wide as that of DSDV and CGSR, greatly improving accuracy of routing information. However, overhead in c-DSDV is similar to that of DSDV because only clusterheads initiate UREQ messages. Delivery ratio increases by about 32~50%.

Ⅰ. 서 론

이동 Ad Hoc 네트워크는 고정된 통신 인프라가 없는 곳에서 이동이 자유로운 무선 노드들이 필요에 따라 임시로 구성하는 네트워크이며, 서로 전송 범위 내에 있지 않는 노드들이 통신을 할 수 있도록 다중 홉 라우팅 방식을 제공한다. 따라서, 모든 노드는 다른 노드가 보내는 패킷을 전달할 수 있는

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음.
* 울산대학교 컴퓨터정보통신공학부 UbiCom 연구실 (hoonoh@ulsan.ac.kr)
논문번호 : KICS2007-02-081, 접수일자 : 2007년 2월 22일, 최종논문접수일자 : 2007년 5월 25일

라우터 기능을 가지고 있다.

일반적으로 이동 Ad Hoc 네트워크는 노드 수, 노드 밀도, 세션 수, 네트워크 디멘션 등 상황의 변화에 대한 제한을 두지 않는다. 따라서, 통신 환경의 변화에 잘 적응할 수 있는 확장성을 갖는 라우팅 프로토콜이 요구된다. 라우팅 프로토콜은 크게 두 가지로 구분된다. 하나는 요구기반 라우팅 프로토콜[2, 6, 8]이며, 다른 하나는 테이블기반 라우팅 프로토콜[7]이다. 요구기반 라우팅 프로토콜은 통신이 필요한 경우에만 경로 탐색을 통해서 경로를 형성하기 때문에 불필요한 네트워크 오버헤드를 줄일 수 있지만 경로 탐색과정에서 플러딩으로 인한 일시적인 네트워크 불안정을 초래하고 경로 설정을 위한 지연 시간이 발생한다. 반면에, 테이블기반 라우팅 프로토콜은 실제 통신의 발생과는 관계없이 모든 노드들 간에 최신의 경로를 유지함으로써 불필요한 통신 자원을 낭비하는 단점이 있다.

DBF 프로토콜[1]을 개선한 테이블 기반 라우팅 프로토콜인 DSDV[7]에서는 각 노드는 개개의 잠재적인 목적지 노드에 대하여 "목적지 노드까지 가기 위한 최단 경로상의 다음 노드"와 "홉 거리"로 구성되는 벡터 테이블을 유지한다. 각 노드는 자신의 라우팅 테이블의 변경된 부분 혹은 전체를 지정된 갱신 주기에 맞추어 이웃 노드들에게 전송하도록 하여 최신의 경로 정보를 유지하고자 한다. 이런 식의 반복적인 갱신에 의해 변경 정보가 전체네트워크로 수렴된다. 따라서, 토폴로지 변화의 갱신 요구에 대한 수렴 속도가 느리기 때문에 라우팅 테이블 정보의 정확도가 떨어지는 단점이 있다.

DSDV의 확장성을 개선하기 위하여 클러스터링 기법을 사용하는 CGSR 이 제안되었다[3]. 모든 노드가 토폴로지 변화에 대한 갱신 요청 메시지를 이웃 노드들에게 만 전송하고, 수신 노드의 갱신 주기에 맞추어 갱신 메시지가 점차적으로 네트워크 전역으로 수렴된다는 점에서 DSDV와 CGSR은 유사하다. 하지만, CGSR은 라우팅 테이블의 엔트리로 클러스터헤드들 만을 사용하기 때문에 라우팅 테이블의 크기는 크게 감소된다. 그러나 노드 이동성이 높은 환경에서 CGSR 프로토콜이 요구하는 "…-클러스터헤드-브릿지-클러스터헤드-브릿지-…"와 같은 클러스터 구조를 모든 이웃 클러스터들 사이에 유지하기가 매우 어렵다. 왜냐 하면, 특정 노드의 이동으로 (…-클러스터헤드-브릿지-브릿지-클러스터헤드-…)의 클러스터 구조가 불가피하게 형성되기 때문이다. 따라서, CGSR에서는 요구하는 구조가 깨어지는 즉시 클러스터링 알고리즘을 다시 실행해야 하고 모든 클러스터헤드의 라우팅 테이블을 재구성해야 하는 문제가 있다.

본 논문에서는 이러한 수렴속도 및 테이블 크기 문제를 동시에 개선하고 또한 복수 경로를 유지함으로써 라우팅 프로토콜의 안정성을 크게 개선하는 클러스터 기반의 c-DSDV라우팅 프로토콜을 제안한다. 각 클러스터에는 멤버들을 관리하는 클러스터헤드가 존재하고 클러스터헤드들끼리 하나의 상위 계층 네트워크 백본을 형성하게 되는데 네트워크 백본에 수정된 DSDV프로토콜을 적용한다. 클러스터헤드들은 갱신 메시지를 이웃 클러스터헤드들과 교환함으로써 전역 라우팅 테이블을 구성한다. 따라서, 라우팅 테이블은 전체 클러스터헤드의 수에 해당되는 엔트리의 수를 가지게 되고 결과적으로 라이팅 테이블의 크기가 그 만큼 줄어든다. 이웃 클러스터헤드들 사이에 갱신 요청 메시지가 플러딩의 형태로 이동할 때 이웃 클러스터헤드들 사이에는 복수 지역 경로가 설정된다. 이웃 클러스터헤드들 사이의 데이터 패킷 이동은 이 지역경로를 사용한다.

갱신 요청 메시지는 플러딩을 통해서 최대 3홉 거리에 있는 이웃 클러스터헤드들까지 전달되기 때문에 하나의 갱신 요청 메시지에 의한 토폴로지 변경의 수렴범위가 DSDV 또는 CGSR 의 그것 보다 최소 아홉 배가 증가한다. 따라서, 라우팅 정보의 정확도가 크게 증가한다. 하지만, 클러스터헤드들만 갱신 요청 메시지를 생성하기 때문에 오버헤드는 DSDV의 그것과 비슷하다. 제안하는 c-DSDV에서는 클러스터헤드들이 생성 및 전송하는 갱신 요청 메시지를 브릿지는 즉시 중계하기 때문에 최대 3홉까지 이동하는데 걸리는 시간은 최악의 경우 (갱신 주기 + 이웃 클러스터헤드 간 전송시간)에 해당되는 지연 시간이 발생할 뿐이다. 브릿지는 수신된 전송 요청 메세지를 바로 전달하기 때문에 이웃 클러스터헤드간 전송시간은 무시할 만 큼 작다고 할 수 있다. 하지만, CGSR의 토폴로지 변경 정보 수렴시간은 최악의 경우 (3 * 갱신 주기)가 된다.

c-DSDV 프로토콜의 성능을 평가하기 위하여 DSDV 프로토콜과 비교하였다. CGSR의 경우는 근본적인 문제점을 가지고 있어 비교를 제외한다. 라우팅 테이블 정보의 정확도 향상으로 인하여 패킷 전송률이 DSDV에 비해 전반적으로 32% ~ 50%정도 크게 개선되었음을 확인할 수 있었다.

논문의 구성은 다음과 같다. 2장에서는 c-DSDV 라우팅 프로토콜을 자세히 기술한다. 3장에서는 시

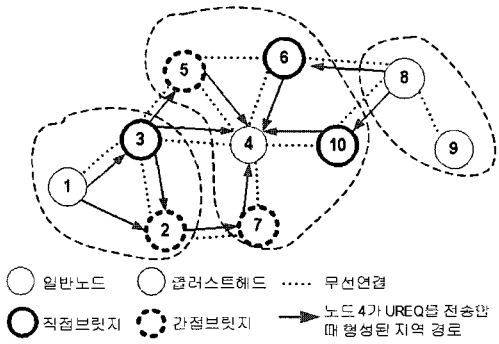

그림 1. 네트워크 클러스터의 예

뮬레이션을 통하여 제안하는 프로토콜의 성능을 평가한다. 그리고 4장에서 결론을 맺는다.

Ⅱ. c-DSDV 라우팅 프로토콜

2.1 클러스터의 형성 및 관리

클러스터링 알고리즘은 클러스터 형성 모듈과 클러스터 관리 모듈로 구성된다. 클러스터 형성은 처음에 한 번 만 실행되고 클러스터 관리를 통해서 최적의 클러스터링을 유지할 수 있도록 한다. 본 논문에서는 Lowest ID 분산 클러스터링 알고리즘[4]을 사용하여 초기 클러스터를 형성한다. 하지만, 초기 클러스터링 알고리즘은 어떤 종류의 클러스터를 사용하든 프로토콜의 성능에 영향을 미치지 않는다. 그림 1은 초기 클러스터를 보여준다.

노드는 *일반노드*, *클러스터헤드*, *직접브릿지*, *간접브릿지* 등 4가지 노드*유형*으로 분류된다. 클러스터헤드는 자신이 속한 클러스터 내의 다른 모든 노드와 1-홉 거리에 있으며 클러스터를 대표하며 멤버 노드들의 정보를 관리한다. 직접브릿지는 두 개의 이웃하는 클러스터헤드를 직접 연결하는 노드이다. 간접브릿지는 다른 직접 혹은 간접 브릿지를 통해서 두 개 이상의 클러스터헤드를 연결하는 노드이며 일반노드는 앞의 세 가지 유형에 속하는 않는 멤버 노드이다.

각 노드는 클러스터를 구성하기 위해서 이웃 노드들의 정보를 관리한다. 이를 위하여 각 노드는 일정한 시간간격, Hello-Interval 마다 이웃 노드들에게 Hello 메시지를 방송한다. Hello 메시지의 구조는 (노드ID, 클러스터헤드ID, 노드유형)으로 되어 있다. Hello-Interval의 2배수의 시간만큼 특정 이웃 노드로부터 Hello 메시지를 받지 못한다면 그 이웃 노드와의 링크가 끊어진 것으로 간주한다.

그림 1은 10개의 노드로 구성된 소규모의 이동 Ad Hoc 네트워크를 보여준다. 점선은 노드 간의 연결상태를 나타내며 Lowest-ID 클러스터링 알고리즘에 의해 구성된 세 개의 클러스터, {1, 2, 3}, {4, 5, 6, 7, 10}, 그리고 {8, 9}를 보여준다.

2.2. 경로 설정

c-DSDV은 두 가지 종류의 경로를 설정한다. 하나는 전역경로이고 다른 하나는 지역경로이다. 전역경로는 클러스터헤드들이 형성하는 백본 네트워크의 경로를 말하며, 각 클러스터헤드는 모든 다른 (잠재적인) 목적지 클러스터헤드에 대하여 라우팅 정보를 유지한다. 지역경로는 이웃하는 클러스터의 클러스터헤드들 사이에 설정되는 경로이다. 다음은 전역경로와 지역경로의 설정 방법에 대하여 기술한다.

표 1. 그림 1에서 노드 8의 전역 라우팅 테이블

목적지 CH*	다음 CH	거리	멤버	일련번호
1	4	4	2, 3	110
4	4	2	5, 6, 7, 10	116
8	8	0	9	114

*CH : 클러스터헤드

2.2.1 전역 경로 설정

각 클러스터헤드는 네트워크 내의 잠재적인 목적지들인 다른 클러스터헤드들에 대한 경로를 유지하기 위하여 전역 라우팅 테이블을 관리한다. 전역 라우팅 테이블의 각 엔트리는 잠재적인 목적지 클러스터헤드, 다음 클러스터헤드, 목적지 클러스터헤드까지의 홉 거리, 목적지 클러스터헤드의 멤버, 그리고 목적지클러스터헤드가 마지막으로 갱신 요청을 한 메시지의 일련번호 등 다섯 개의 필드로 구성된다. 표 1에는 그림 1의 토폴로지가 안정 상태에 있을 때 클러스터헤드 8이 가지게 되는 전역 라우팅 테이블의 예를 나타낸다. 클러스터헤드 8이 목적지 클러스터헤드 1에게 패킷을 전송하려면 패킷을 보낼 다음 클러스터헤드는 4이며 목적지 클러스터헤드까지의 거리는 4라는 것을 보여준다. 그리고, 클러스터헤드 1은 2개의 멤버 2, 3을 가지고 있다는 것을 알 수 있다. 일련번호 (예를 들면, 110)는 해당 클러스터헤드가 갱신 요청 메시지를 보낼 때 마지막으로 생성한 번호이다.

전역 라우팅 테이블을 최신으로 유지하기 위하여 각 클러스터헤드는 주기적으로 혹은 전역 라우팅 테이블의 내용이 변경되면 갱신 요청 메시지 (UREQ)

표 2. UREQ 메시지 형식

필드	의미
ureq_src	UREQ 를 생성한 소스 노드 주소
ureq_src_seq	소스 노드가 생성한 일련 번호
ureq_id	UREQ 메세지의 고유 번호
ureq_fowarde r	현재 수신자에게 UREQ 메세지를 전달한 노드 주소 (즉, 전달하는 노드 주소)
hop_count	UREQ가 이동한 홉 수
reoute_info	UREQ가 운반하는 UREQ 소스 노드의 라우팅 테이블 정보

를 이웃 클러스터헤드들에게 전송한다. 갱신요청 메시지 UREQ은 6개의 필드로 되어 있으며 그 형식은 표 2와 같다.

UREQ 메시지 형식에서 ureq_src는 UREQ를 생성 및 전송한 클러스터헤드의 주소이며, ureq_src_seq는 ureq_src가 생성한 일련번호로써 정상적인 경우에는 전송시 마다 2씩 증가한다. ureq_id는 해당 UREQ 메세지의 고유 번호이며, ureq_forwarder는 UREQ 메시지를 전달하는 노드이며 UREQ 메시지를 수신한 노드는 자신의 주소로 변경하여 방송한다. hop_count는 소스 클러스터헤드로부터 현재 노드까지 이동한 거리이며 이웃 클러스터헤드 사이의 최대 이동 거리는 3-홉이다 [5]. route_info는 전역 라우팅 테이블의 엔트리들 중에서 이전 갱신 요청 메시지 전송 이후에 변경된 부분만을 포함한다. 클러스터헤드들은 이웃 클러스터헤드들과 UREQ 메시지의 교환을 통해서 결국 네트워크 내의 모든 클러스터헤드들에 대한 토폴로지 정보를 유지하게 된다.

UREQ 메시지의 불필요한 재전송을 방지하기 위하여 다음 3가지 조건을 동시에 만족시키는 노드만이 갱신 요청 메시지를 재전송한다.

- 노드 타입이 브릿지인 경우;
- *hop_count*가 3보다 작은 경우; 그리고
- 메시지를 처음 수신한 경우.

특정 UREQ메시지가 처음으로 수신된 것인지를 검사하기 위하여 <ureq_src, ureq_id>를 사용한다. ureq_src_seq 는 DSDV에서처럼 경로 루프 문제를 해결하기 위하여 사용한다. 정상적인 갱신인 경우에는 일련번호를 2씩 증가한다. 노드 이동이나 파워 문제로 인하여 링크 파손이 발생한 경우에 임의의 타임아웃 시간 동안 기존의 이웃 클러스터헤드로부터 갱신 요청 메시지를 수신하지 못하는 경우가 발생한다. 경로 파손을 인지한 클러스터헤드는 해당 이웃 클러스터헤드에 대한 거리를 ∞ 값으로 변경하고 끊어진 지역 경로를 이웃 클러스터헤드들에게 알려주기 위하여 새로운 갱신 요청 메시지를 생성하여 방송한다. 끊어진 클러스터헤드에 대한 일련번호를 1만큼 강제로 증가하여 전송함으로써 수신하는 다른 클러스터헤드들이 즉각 파손된 경로 정보를 전역 라우팅 테이블에 반영할 수 있도록 한다.

2.2.2 지역 경로 설정

UREQ 메시지가 이웃 클러스터헤드 사이를 이동하는 동안에 UREQ를 수신하는 모든 노드는 UREQ를 전달한 노드(즉, ureq_forwarder)에 대하여 포인터를 생성하게 된다. 이렇게 함으로써 모든 UREQ 메시지를 수신하는 노드들은 그 ureq_src에 대한 경로를 설정하게 되고 이러한 경로들을 지역 경로라고 한다. 지역 경로는 타임아웃 값을 설정함으로써 시간이 지나면 해제된다. 그림 1의 화살표는 클러스터헤드 4가 UREQ메시지를 이웃 클러스터헤드들에게 방송하는 경우 클러스터헤드 4를 향해 역으로 설정된 모든 지역경로를 나타낸 것이다.

2.3 경로 유지 및 보수

노드의 이동으로 인하여 네트워크 토폴로지가 변경되는 경우에 클러스터링 알고리즘을 다시 수행하지 않는다. 변경된 토폴로지 부분에 대하여 클러스터 정보 및 전역 라우팅 테이블 정보를 변경시키고, UREQ메시지를 생성 및 송신하여 다른 이웃 클러스터들이 갱신을 할 수 있게 한다. 따라서, c-DSDV 프로토콜에서는 클러스터 구조 변경은 전역 라우팅 테이블의 갱신을 동반하고 UREQ 메시지의 생성 및 전송을 유발한다. 따라서, 네트워크 오버헤드 문제를 효과적으로 다루기 위하여 클러스터 구조를 변경시키는 요인을 살펴본다.

- 첫째는 두 클러스터헤드가 인접하여 한 클러스터헤드가 자신의 역할을 포기하고 이웃 클러스터헤드의 멤버 노드가 되는 경우이다. 이렇게 하는 이유는 클러스터헤드의 수가 적으면 적을수록 전역 라우팅 테이블의 엔트리 수가 적어지기 때문이다.

- 둘째는 멤버 노드가 이동으로 인하여 자신의 클러스터헤드의 전송 범위를 벗어나는 경우이다. 이 경우에는 해당 멤버 노드가 다른 클러스터헤드의 멤버가 되거나 자신이 스스로 클러스터헤드가 될 수 있다.

기본적으로, 새로운 멤버를 갖는 클러스터헤드는

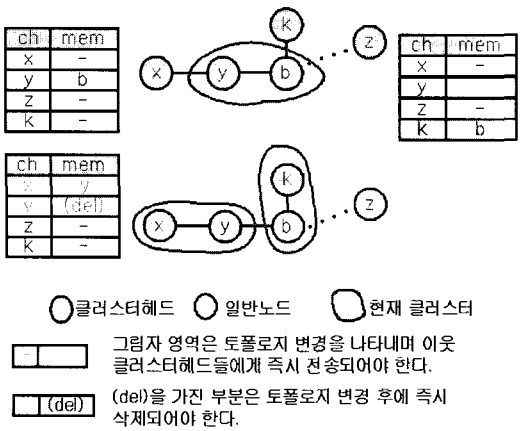

그림 2. 두 클러스터헤드가 다른 하나의 멤버가 되는 예

이로 인하여 변경된 전역 라우팅 테이블의 부분을 이웃 클러스터헤드들에게 전송해야하고, 새로 클러스터헤드가 된 노드는 자신의 전역 라우팅 테이블을 생성하여 이웃 클러스터헤드들에게 전송해야 한다. 첫 번째 경우에 어떤 클러스터헤드가 자신의 역할을 포기하고 인접한 클러스터헤드의 멤버가 되는 경우에 자신의 기존 멤버들은 모두 다른 클러스터헤드에 가입하여야 하는 문제가 발생한다. 이 때 기존의 멤버의 수가 많고 그들이 여러 클러스터헤드에 가입하는 경우에 새로운 멤버를 갖는 클러스터헤드들은 모두 UREQ 메시지를 전송해야 한다. 이 경우에 과도한 컨트롤 오버헤드를 발생시키기 때문에 통제할 필요가 있다. 인접한 두 클러스터헤드는 다음 두 가지 조건을 모두 만족할 때만이 하나가 다른 클러스터헤드의 멤버로 전환될 수 있다:

(1) 모든 멤버들은 직접브릿지이다; 그리고
(2) 멤버 수가 ΔM보다 작거나 같다.

여기서 ΔM은 네트워크 전개 환경에 따라 정해지는 임의의 작은 정수 값이다. 첫 번째 조건은 어떤 클러스터헤드가 다른 클러스터헤드의 멤버가 된다 할지라도 그 클러스터헤드의 멤버들이 모두 직접브릿지 이기 때문에 즉시 다른 이웃 클러스터헤드에 가입할 수 있다는 것을 의미한다. 첫 번째 조건의 예를 그림 2에 도시하였다. 클러스터헤드 y의 하나의 직접브릿지 b를 멤버로 가지고 있기 때문에 클러스터헤드 y는 클러스터헤드의 역할을 포기하고 이웃 클러스터헤드 x의 멤버 노드가 될 수 있다. 이 경우 b는 쉽게 이웃 클러스터헤드의 멤버가 될 수 있다. 두 번째 조건은 (1)의 조건이 만족하더라도 멤버수가 많으면 그 멤버 노드들이 다른 클러스터에 가입하고 UREQ 메시지를 유발함으로써 발생하는 오버헤드가 커지므로 그 수를 제한한다는 것을 의미한다. ΔM의 값을 작으면 제어 메시지 오버헤드는 줄어드나 클러스터의 수는 증가할 수 있다.

이러한 조치에도 불구하고, 토폴로지의 변화가 일어날 때마다 즉시 클러스터정보를 변경하고 전역 라우팅 테이블을 갱신하기 위하여 UREQ를 전송하는 경우에 일시적인 갱신 메시지의 증가로 인하여 컨트롤 오버헤드가 크게 증가하는 현상이 발생할 수 있다. 이를 제어하기 위하여 클러스터헤드는 UREQ를 한 번 전송하고 나서 다음 UREQ 전송할 때까지 최소한의 시간 경과를 갖도록 제한한다. 이 시간 간격을 UREQ 갱신 간격의 하위 한계값, 즉 *UREQ-LB-UpdateInterval* 이라고 한다. 이 값을 사용하여 클러스터헤드는 *UREQ-LB-UpdateInterval* 내에 한 번만 UREQ를 전송할 수 있도록 제한한다. 따라서 클러스터헤드는 이 시간 간격 동안 갱신된 전역 라우팅 테이블 정보를 축적하여 한 번에 갱신하게 된다.

2.4 데이터 전송

데이터 전송은 전역 라우팅 테이블과 지역 라우팅 테이블을 이용한다. 패킷을 전송할 노드는 우선 자신의 클러스터헤드에게 패킷을 전송한다. 클러스터헤드는 전역 라우팅 테이블을 검색하여 먼저 목적지 노드가 속한 목적지 클러스터헤드를 찾는다. 그 다음, 클러스터헤드는 전역 라우팅 테이블로부터 목적지 클러스터헤드에 대한 다음 클러스터헤드를 얻는다. 다음 클러스터헤드까지의 전송은 지역경로를 따른다. 이렇게 반복 수행을 통해서 패킷이 목적지 클러스터헤드에 도달하게 되고 목적지 클러스터헤드는 멤버인 목적지 노드에게 패킷을 전송한다.

그림 1에서 노드 9가 노드 3에게 패킷을 전송하는 예를 보자. 노드 9는 자신의 클러스터헤드 8에게 패킷을 전송한다. 수신 클러스터헤드 8은 자신의 전역 라우팅 테이블 (표1)을 참조하고 목적지 노드 3이 클러스터헤드 1의 멤버임을 알게 된다. 노드 1이 목적지 클러스터헤드가 된다. 따라서, 클러스트헤드 8은 패킷을 목적지 클러스터헤드 1까지 전송해야 한다. 표1의 전역 테이블을 참조하면 노드 4가 목적지 클러스터헤드 1을 향해서 가는 다음 클러스터헤드임을 알 수 있다. 지역 경로 (8, 6, 4) 및 (8, 10, 4)는 이미 설정되어 있으므로 이 중 하나의 지역 경로를 따라 4에게 패킷을 전송한다. 패킷을 수신한 클러스터헤드 4는 자신의 전역 라우팅 테이

블을 참조하고 목적지 클러스터헤드1이 바로 다음 클러스터헤드임을 알 수 있을 것이다. 4에서 1까지의 지역 경로들은 이미 설정되어 있기 때문에 최단 경로인 (4, 3, 1)을 따라 전송한다. 도중에 패킷을 수신한 노드 3은 자신이 목적지라는 것을 알고 패킷을 재전송하지 않고 처리 완료한다.

Ⅲ. 성능 평가

GlomoSim 2.03[9] 시뮬레이터를 사용하여 제안한 c-DSDV프로토콜을 대표적인 테이블 기반 방식의 프로토콜인 DSDV와 성능을 비교하였다. 표2와같이 고정 파라메타 값을 사용하였으며, 노드 이동 속도와 노드 위치 이동 후 정지 시간을 변경시키면서 성능을 측정하였다.

그림 3은 정지 시간 변화에 대한 전송률을 나타내었다. 범례 x-y는 x는 프로토콜 명, y는 최대 이동 속도를 나타낸다. 노드 최대 이동 속도가 5m/s인 경우에 거의 균일하게 28%의 개선율을 보였으며, 노드 최대 속도가 20m/s인 경우에 정지시간의 값에 따라 대체로 32~50%의 개선율을 보였다. 이동성이 높은 경우에 그 차이가 커짐을 알 수 있다.

이러한 차이의 가장 큰 요인은 c-DSDV프로토콜의 UREQ메시지는 최대 3-홉 거리에 있는 이웃 클러스터헤드들까지 전송됨으로써 토폴로지 변화의 수렴범위가 DSDV에 비해 훨씬 크기 때문이다. DSDV와 CGSR의 수렴범위를 πR^2 (R = 홉 수, 즉 홉 수를 반지름으로 대응시킴)이라고 할 때, c-DSDV의 수렴 범위는 $\pi(3R)^2$이다. 따라서, c-DSDV의 수렴범위는 DSDV의 수렴범위보다 최소 9배에 이른다. 특히, c-DSDV는 클러스터헤드 사이에 다수 지역경로를 설정함으로써 노드 이동성이 높은 경우에도 비교적 안정된 전송을 할 수 있었다.

표 2. 시뮬레이션 파라메타

파라메타	값
노드 이동 패턴	Random Waypoint
노드 수	100
디멘션 크기	1500 x 300
전송 범위	250 m
대역폭	2 Mbps
트래픽 유형	CBR
CBR세션 수	10개 (전송간격: 2sec.)
데이터 크기	512 bytes
시뮬레이션 시간	600 (sec.)

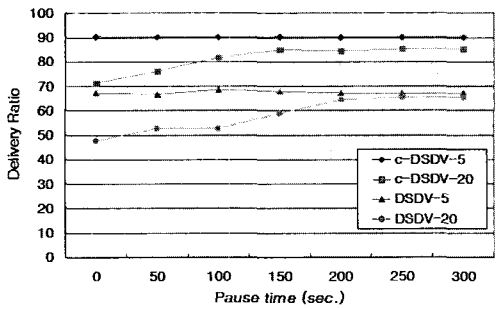

그림 3. 전송률 비교

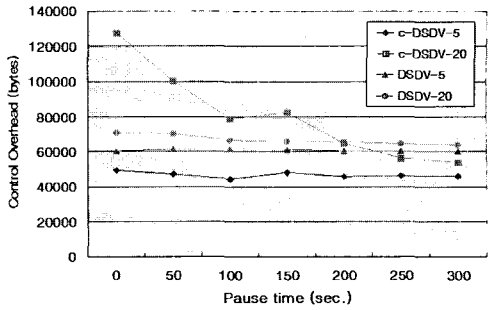

그림 4. 컨트롤 오버헤드 비교

그림 4은 컨트롤 메시지의 오버헤드를 보여준다. DSDV의 경우에는 라우팅 테이블 교환으로 인한 오버헤드를 나타내며, c-DSDV의 경우에는 Hello메시지와 UREQ를 포함한다. 특히, c-DSDV의 경우에 클러스터 유지를 위해서 별도의 메시지를 사용하지 않는 대신에 UREQ및 Hello메시지의 빈도가 증가한다. 이 두 콘트롤 오버헤드를 산출에 반영하였다. 또한, c-DSDV의 경우에는 클러스터헤드만 UREQ를 생성하여 전송하지만 중간 브릿지들은 그 메시지를 전달하므로, 전달로 인한 오버헤드를 포함하였다. 그럼에도 불구하고 노드 이동성이 비교적 낮은 경우에는 c-DSDV의 오버헤드가 전반적으로 낮게 나타나고 있다. 하지만, c-DSDV의 경우에 이동성이 크게 증가하는 경우에 클러스터 유지를 위

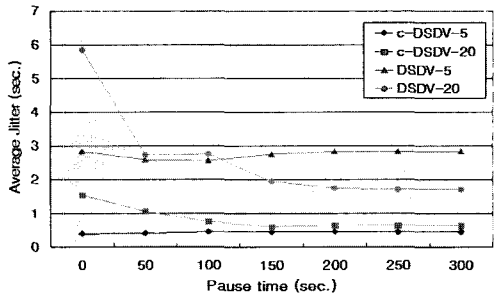

그림 5. 지터 비교

하여 Hello 및 UREQ메시지의 수가 증가하고 메시지 오버헤드가 증가한다는 것을 알 수 있다. 이러한 결과는 c-DSDV의 수렴범위 향상과 결합하여 전송율을 향상 시킨다는 것을 알 수 있다 (그림 3참조).

그림 5에서 c-DSDV는 노드 이동 속도에 무관하게 DSDV에 비교하여 매우 낮은 평균 지터를 보인다. 이는 c-DSDV의 사용으로 경로의 안정성이 개선되었음을 의미한다.

Ⅵ. 결 론

DSDV를 기반으로 제안한 c-DSDV프로토콜은 전역 라우팅 테이블이 클러스터헤드만을 엔트리로 구성함으로써 갱신 요청 메시지에 포함되는 경로 테이블의 크기를 크게 줄였다. 거의 동일한 오버헤드를 유지하면서 DSDV와 CGSR보다 갱신 메시지의 수렴범위를 9배로 확장하였으며, 그 결과 라우팅 정보의 정확성이 크게 증가하였다. 또한 클러스터헤드간에 복수의 지역 경로를 설정 및 유지하기 때문에 추가적인 오버헤드 없이 링크 손실에 대해 빠른 복구를 제공한다.

c-DSDV는 모든 패킷들이 목적지까지의 최단 경로 상에 있는 클러스터들의 클러스터헤드를 경유하기 때문에 클러스터헤드들의 혼잡현상이 발생할 수 있다. 또한, 최단 경로를 따라 패킷이 이동하지 못한다. 이러한 문제를 보완하기 위하여 클러스터헤드 혼잡 방지 방법 및 경로 최적화 기법에 대한 추후 연구가 필요하다.

참 고 문 헌

[1] D. Bertsekas and R. Gallager, Data Networks, Second Edition, Prentice Hall, Inc., 1992.

[2] D. B. John and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in mobile computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[3] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," *The IEEE Singapore International Conference on Networks*, pp. 197-211, IEEE, 1996

[4] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 7, pp. 1265-1275, 1997.

[5] H. Oh and H. S. Park, "Communication architecture and protocols for broadcast-type mobile multimedia Ad Hoc networks," *2002 Military Communications Conference*, Sponsored by IEEE, pp. 1-6, Oct. 2002.

[6] V. D. Park and M. S. Corson, "A Highly adaptive distributed routing algorithm for mobile wireless networks," *IEEE INFOCOM '97*, Kobe,Japan,1997.

[7] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance- vector routing (DSDV) for mobile computers," *ACM SIGCOMM: Computer Communications Review*, vol. 24, no. 4, pp.234-244, Oct. 1994.

[8] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," *Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, Feb. 1999.

[9] UCLA Parallel Computing Laboratory and Wireless Adaptive Mobility Laboratory. GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems, http://pcl.cs.ucla.edu/projects/glomosim

오 훈 (Hoon Oh) 정회원

1981년 성균관대학교 전자공학 학사

1993년 텍사스A&M대학교 전산학 석사

1995년 텍사스A&M대학교 전산학 박사

1996년 삼성전자 중앙연구소 수석연구원

2005년~현재 울산대학교 컴퓨터정보통신공학부 조교수

<관심분야> 실시간 시스템, 임베디드 시스템, 애드 혹 및 센서 네트워크 프로토콜