

# c-DSDV 라우팅 프로토콜에서 클러스터헤드의 혼잡 회피를 위한 적응적 라우팅 방법

정회원 오 훈\*, 윤 석 열, 준회원 부 초우 투완\*

## Adaptive Routing Scheme to Avoid Clusterhead Congestion in c-DSDV Routing Protocol

Hoon Oh\*, Seok Yeol Yun\* *Regular Members*, Trong Tuan Vu\* *Associate Member*

요 약

DSDV의 확장성을 개선하기 위해 제안된 클러스터 기반의 c-DSDV 라우팅 프로토콜은 모든 클러스터헤드들이 클러스터헤드들만을 잠재적 목적지로 하는 라우팅 테이블을 관리하고, 이를 최신으로 유지하기 위해서 각 클러스터헤드는 갱신 요청 메시지를 주기적으로 혹은 토폴로지 변경 시에 이웃 클러스터헤드들에게 플러딩한다. 따라서, 하나의 갱신 메시지에 의한 토폴로지 수렴범위가 9배 이상 확장되어 라우팅 정확도가 크게 개선되었다. 하지만, 전송되는 패킷들이 경로 상의 모든 클러스터헤드를 경유함으로써 클러스터헤드의 혼잡이 발생한다. 이러한 문제를 개선하기 위하여 패킷 전송 중에 클러스터헤드를 우회할 수 있는 지를 판단할 수 있는 적응적 라우팅 방법을 제안한다. 그 결과, 전송 경로가 단축됨은 물론, 클러스터헤드의 혼잡 감소에 의한 큐의 단축으로 인하여 종단간 지연시간이 크게 개선되었다. 시뮬레이션을 통해서 약 40% 이상 지연 시간이 개선됨을 보였다.

**Key Words** : Ad Hoc Networks, DSDV, Adaptive, Cluster, Clusterhead Congestion

ABSTRACT

In the c-DSDV routing protocol proposed to improve the scalability of DSDV, clusterheads manage a routing table that has only clusterheads as potential destinations and flood update request message to its neighbor clusterheads periodically or at the time of topology change. Accordingly, the convergence range of topology change by a single update request message was expanded nine times as wide as that of DSDV, increasing routing correctness; however, c-DSDV suffers from the congestion of clusterheads since data packets always go through clusterheads of the clusters on the routing path. To improve this problem, we propose an adaptive routing scheme that judges if detouring clusterhead is possible on the fly while packets are forwarded. As a result, a routing path length is shortened and an end-to-end delay is improved by the reduced queue length. It shows that the end-to-end delay is reduced by almost 40% through simulation.

### I. 서 론

이동 애드 혹 네트워크는 인프라구조가 없는 장소에서 이동이 자유로운 무선 이동 노드들로 구성

되는 임시 네트워크이다. 서로 전송 범위 내에 위치하지 않은 노드 간의 통신을 위하여 다중 홉 방식을 사용하기 때문에 모든 이동 노드는 라우터 기능을 갖는다. 이러한 동적 토폴로지 및 다중 홉 라우

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업의 연구결과로 수행됨 (IITA-2008-(C1090-0801-0039))

\* 울산대학교 컴퓨터정보통신공학부 UbiCom 연구실 (hoonoh@ulsan.ac.kr)

논문번호 : KICS2008-01-003, 접수일자 : 2007년 11월 13일, 최종논문접수일자 : 2008년 2월 13일

팅의 특성 외에 이동 애드 혹 네트워크는 응용 분야에 따라 노드의 수, 노드의 이동성, 노드의 밀도, 네트워크 전개 영역의 크기 등이 변한다. 따라서, 확장성과 신뢰성을 고려한 라우팅 프로토콜을 설계하는 것은 대단히 어려운 문제이다.

확장성 문제를 해결하기 위한 다양한 클러스터링 알고리즘 및 계층적 접근방식이 제안되었다<sup>1,2,3,5,7)</sup>. 클러스터링 알고리즘을 적용하는 방식에서는 전체 네트워크가 서로 중첩되지 않는 클러스터들로 표현된다. 각 클러스터는 클러스터에 속하는 멤버와 클러스터 외부를 연결하는 노드들의 정보를 관리하는 클러스터헤드를 가진다. 이 경우에 패킷 라우팅이 실행되는 도중에 패킷들이 클러스터헤드를 통과할 확률이 높다. 그 결과 클러스터헤드의 부하 증가로 인한 전력 소모가 증가하고 클러스터헤드 혼잡으로 인한 패킷 전송 지연 현상이 발생한다.

본 논문에서는 DSDV 프로토콜<sup>6)</sup>의 확장성을 개선하기 위해 제안된 c-DSDV<sup>5)</sup>의 클러스터 헤드 혼잡 현상을 개선하기 위한 적응적 라우팅 방식을 제안한다. 설정된 경로를 따라 데이터 패킷을 전송하는 중에 패킷을 가진 노드는 전송할 다음 노드가 클러스터헤드이면 이를 우회할 수 있는지를 판단한다. 시뮬레이션을 통해서 제안된 방식이 얼마나 효과적인가를 평가하였다. 그 결과, 경로의 길이 단축은 물론 클러스터헤드의 혼잡을 개선함으로써 종단간 지연시간이 40% 이상 개선되었다.

논문의 구성은 다음과 같다. II장에서는 연구 배경을 설명하고, III장에서는 적응적 라우팅 방법을 설명한다. IV장에서는 시뮬레이션을 통하여 성능을 평가하고, V장에서 결론을 맺는다.

## II. 배경

### 2.1 c-DSDV 및 문제점

c-DSDV 프로토콜에서는 클러스터링 알고리즘을 통해서 구성된 클러스터들의 클러스터헤드들이 전체 네트워크의 상위 계층인 네트워크 백본을 형성한다. 각 클러스터헤드는 잠재적 목적지인 각 클러스터헤드에 도달하기 위한 다음 클러스터헤드, 목적지 클러스터헤드까지의 거리, 목적지 클러스터헤드가 관리하는 멤버 등을 포함하는 GRIT (Global Routing Information Table)을 관리한다. 토폴로지가 변화에 대하여 GRIT를 최신으로 유지하기 위하여 각 클러스터헤드는 주기적으로 혹은 토폴로지 변경을 감지할 때 갱신된 부분의 정보를 포함하는 UREQ

(Update Request Message)를 자신의 이웃 클러스터헤드들에게 플러딩 한다. 네트워크 상의 모든 노드들은 이웃 클러스터헤드들 사이에 이동하는 UREQ를 이용하여 LRIT (Local Routing Information Table)을 관리한다. UREQ를 수신하는 모든 노드는 UREQ를 생성한 클러스터헤드까지 다음 노드와 최단 거리를 저장할 수 있다.

GRIT와 LRIT이 구성되면, 패킷을 송신하고자 하는 노드는 먼저 자신의 클러스터헤드에게 전송한다. 이를 수신한 클러스터헤드는 자신의 GRIT로부터 목적지 노드가 속한 목적지 클러스터헤드를 찾고 목적지 클러스터헤드로 가기 위한 다음 클러스터헤드를 얻는다. 그리고 LRIT에 있는 지역 경로를 이용하여 다음 클러스터헤드로 패킷을 전송한다. 패킷을 수신한 다음 클러스터헤드는 위와 동일한 과정을 반복함으로써 패킷은 목적지 클러스터헤드에 도달되고, 계속해서 목적지 노드로 전달된다.

c-DSDV 프로토콜은 우수한 성능을 나타냄에도 불구하고 패킷들이 목적지로 가는 도중에 통과하는 모든 클러스터들의 클러스터헤드를 경유함으로써 클러스터헤드의 혼잡이 발생하고 종단간 패킷 지연시간을 증가시킨다. 특히, 세션의 수가 증가할수록 클러스터헤드 혼잡 현상이 심화된다.

### 2.2 문제 해결방안

그림 1에서 c-DSDV 프로토콜이 소스로부터 목적지까지 패킷을 전달하는 과정을 살펴보자. 클러스터헤드들만이 글로벌 라우팅 테이블을 관리하고 있기 때문에 모든 전송 패킷들은 클러스터헤드로 전달되어야 라우팅을 수행할 수 있다. 예를 들면, 그림 1에서 클러스터헤드 x에 속하는 노드 s가 클러스터헤드 y에 속하는 노드 d로 패킷을 전송한다고 하자. 노드 s는 자신의 클러스터헤드 x로 패킷을 전송한다. 전송된 패킷은 지역 경로 (x, a, c, y) 혹은 (x, b, c, y)를 거쳐 다음 클러스터헤드 y로 전달되고, 최종적으로 목적지 d에게 전달된다.

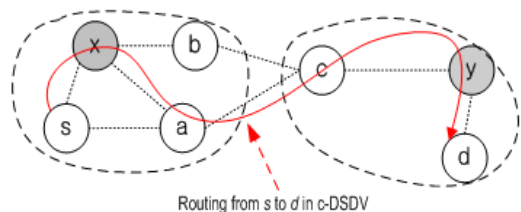


그림 1. 전송 경로의 수정

노드 s가 이웃 클러스터헤드 x의 전역 라우팅 테이블을 공유하고 있다고 가정해 보자. 노드 s는 이 전역 라우팅 테이블을 참조하여 목적지 클러스터헤드로 가는 다음 클러스터헤드가 y인 것을 미리 알 수 있다. 노드 s에서 클러스터헤드 y까지 지역 경로 (s, a, c, y)가 이미 설정되어 있는 경우에, 노드 s는 패킷을 지역경로를 따라 목적지 클러스터헤드 y로 직접 보낼 수 있다. 물론, 소스가 아닌 중간 노드의 경우에도 다음 노드가 클러스터헤드이면 동일한 과정을 통해서 클러스터헤드를 우회할 수 있다.

한편, 노드 c와 목적지 d가 연결되어 있다고 가정하자. 노드 c는 다음에 전달할 노드 y가 목적지 클러스터헤드인지를 알 수 있다면 노드 c는 노드 d가 자신의 이웃인지를 먼저 검사한다. 그렇다면 클러스터헤드 y를 우회하여 목적지 d에게 직접 전달할 수 있다. 이런 식으로 이웃 노드의 정보를 이용하여 목적지 클러스터헤드를 우회할 수 있다.

### III. 적응적 라우팅 방법

3.1 전역 라우팅 테이블 및 지역 라우팅 테이블  
 각 클러스터헤드는 클러스터헤드들 간의 라우팅 정보를 제공하는 GRIT를 관리한다. GRIT는, 각 클러스터헤드에 대하여, 해당 클러스터헤드 주소, 해당 클러스터헤드로 가는 다음 클러스터헤드, 해당 클러스터헤드까지 거리, 해당 클러스터헤드가 관리하는 클러스터 멤버, 해당 클러스터헤드가 마지막으로 생성한 갱신 요청 메시지의 일련번호 등을 관리한다. 일련 번호의 관리 방식은 DSDV와 같다.

그림 2는 20개의 노드가 7개의 클러스터를 형성하고 있는 소규모 네트워크이다. 점선은 노드 연결 상태를 나타내고, 일반노드, 클러스터헤드, 그리고 두 인접한 클러스터를 연결하는 브릿지 등과 같은 노드가 있다. 화살표는 출발점의 노드가 화살표 끝점의 노드에 대한 주소를 가진다는 것을 의미한다.

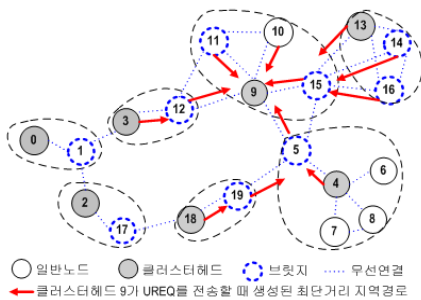


그림 2. 네트워크 클러스터 및 지역 경로설정

표 1. 그림 2에서 GRIT<sub>9</sub>

목적지 CH	다음 CH	거리	멤버	일련번호
0	3	4	1	120
2	3	4	13	110
3	3	2	12	124
4	4	2	5	116
9	9	0	10, 11, 15	114
13	13	2	14, 16	110
18	18	3	19	112

표 2. 그림 2에서 GRIT<sub>13</sub>

목적지 CH	다음 CH	거리	멤버	일련번호
0	9	6	1	120
2	9	6	13	110
3	9	4	12	124
4	9	3	5	116
9	9	2	10, 11, 14	114
13	13	0	14, 16	110
18	9	5	19	112

즉, 노드 18은 노드 19의 주소를 가지고 있다. 파선으로 된 타원은 클러스터 경계를 표시한다. 클러스터헤드 ch의 전역 라우팅 테이블을 GRIT<sub>ch</sub> 라고 표기할 때, 표 1은 GRIT<sub>9</sub>, 표 2는 GRIT<sub>13</sub>을 나타낸다. GRIT를 최신으로 유지하기 위해서 각 클러스터헤드는 주기적으로 혹은 토폴로지 변경 이벤트가 발생할 때 마다 UREQ를 모든 이웃 클러스터헤드들에게 플러딩한다. 네트워크 파티션이 발생하지 않는 한 모든 이웃 클러스터헤드 사이의 거리는 최대 3홉이다<sup>4)</sup>. 따라서, UREQ 메시지는 TTL = 3으로 설정된다. 이 메시지를 수신하는 클러스터헤드 및 일반노드를 제외한 모든 브릿지는 TTL값을 1씩 감소하고 UREQ를 재전송한다.

UREQ 메시지가 이웃 클러스터들을 향해서 이동하는 동안 모든 수신 노드는 UREQ를 보낸 소스 클러스터헤드로 가는 다음 노드 및 소스 클러스터헤드까지의 최단 거리를 LRIT에 관리한다. 그림 2에서 클러스터헤드 9가 UREQ 메시지를 전송할 때 클러스터헤드 9로 가는 최단 거리에 있는 다음 노드와 최단 거리 (3 - TTL)를 저장한다.

**정리1:** c-DSDV에서 연속적으로 연결된 노드 (1) h, k, (2) h, i, k 혹은 (3) h, i, j, k에 대하여 h가 클러스터헤드이고 i, j가 브릿지이면 k에서 h까지 지역경로가 설정된다.

**증명:** c-DSDV에 따르면 클러스터헤드는 UREQ 메시지의 TTL을 3으로 설정하여 전송하며, 브릿지는 수신한 UREQ의 TTL을 1 감소하여 0이 아니면 재전송한다. 먼저 (1)의 경우를 보자. 브릿지 i 및 j

는 목적지  $h$ 로 가는 자신의 선행 노드에 대하여 경로 포인트를 설정하고 거리를 저장한 후 재전송한다. 노드  $k$ 는 UREQ를 수신하고 위와 동일하게 선행 노드  $j$ 에 대한 경로 포인트 및 최단거리를 저장한다. 결과적으로, 경로  $(k, j, i, h)$ 가 설정된다. (1)과 (2)의 경우에 동일하다.

정리 1에 의해서 그림 2에서 노드 15의 LRIT<sub>15</sub>는 표 3과 같다.

표 3. 그림 2에서 LRIT<sub>15</sub>

목적지 CH	다음 노드	거리
4	5	2
9	9	1
13	13	1
18	5	3

### 3.2 전역 라우팅 테이블의 공유 및 관리

노드가 이웃하고 있는 클러스터헤드를 우회하기 위해서는 이웃 클러스터헤드의 GRIT를 공유 및 관리해야 한다. c-DSDV 프로토콜에서 클러스터헤드는 주기적으로 혹은 토폴로지 변경이 발생하는 경우에 UREQ를 최대 3홉 이내에 있는 이웃 클러스터헤드들에게 플러딩하기 때문에 이웃 노드들은 그러한 정보를 수신할 수 있다. 이 때, 이웃 노드는 UREQ의 테이블 내용을 자신이 관리하고 있는 해당 클러스터헤드의 GRIT의 내용과 비교하여 동기화 한다.

노드가 새로운 클러스터헤드를 이웃으로 가지면 처음 UREQ 메시지를 수신할 때 이웃 클러스터헤드에 대한 GRIT를 생성한다. 노드가 특정 이웃 클러스터헤드로부터 UREQ 전송 간격의 두 배의 시간 동안 UREQ를 직접 받지 못하면 링크가 끊어진 것으로 간주하고 해당 클러스터헤드에 대한 전역 라우팅 테이블을 삭제한다.

### 3.3 적응적 라우팅 알고리즘

c-DSDV에서 경로상의 어떤 노드가 다음 두 가지 조건을 동시에 만족하면 다음 노드를 우회할 수 있다:

- 다음 노드가 클러스터헤드이다; 그리고
- 현재 노드로부터 경로 상의 두 번째 클러스터헤드까지 지역 경로가 존재한다.

노드는 이웃인 다음 클러스터헤드의 GRIT를 검색하여 자신으로부터 두 번째 클러스터헤드를 얻을 수 있고, 자신의 LRIT를 검색하여 그 두 번째 클러스터헤드까지 지역경로가 있는지를 알 수 있다.

그림 3에 기술한 적응적 라우팅 알고리즘은 소스가 패킷을 처리하는 부분과 중간 노드가 패킷을 처

리하는 부분으로 나누어져 있다. 데이터 패킷을 가진 노드  $src$ 는 자신이 클러스터헤드이면 자신의 전역 라우팅 테이블 GRIT<sub>src</sub>로부터 다음 클러스터헤드  $n_1$ 을 찾고  $n_1$ 까지 지역경로가 있으면 지역경로를 따라 다음 클러스터헤드  $n_1$ 으로 전송한다. 만일 소스 노드가 클러스터헤드가 아니면 먼저 자신의 클러스터헤드  $n_0$ 의 GRIT<sub>n0</sub>로부터 다음 클러스터헤드  $n_1$ 을 얻고  $n_1$ 까지의 지역경로가 있으면 지역 경로를 따라  $n_1$ 으로 전송하고, 그렇지 않으면 자신의 클러스터헤드  $n_0$ 로 전달한다. 전자의 경우  $n_1$ 이  $src$ 의 다른 1홉 이웃 클러스터헤드일 수도 있다. 이 경우에 다시 GRIT<sub>n1</sub>로 이용하여 위의 과정을 반복한다.

클러스터헤드  $n_0$ 로 향하는 패킷을 수신하는 중간 노드  $x$ 는 자신이 목적지 이면 해당 패킷을 처리한다. 목적지 노드가 아닌 경우에 다음 클러스터헤드  $n_0$ 에 대하여 지역경로 상의 다음 노드  $k$ 를 얻는다. 노드  $k$ 가  $n_0$ 이면 처리과정은 이전 설명과 같다. 그렇지 않으면 노드  $k$ 로 패킷을 전송한다.

```

// chead: clusterhead
At node src that originates data
if src ≠ chead then
    n0 = chead of src;
    find next ch n1 w.r.t dst's chead from GRITn0;
    if there exists a local path from src to n1 then
        if the next node of the local path is n1 then
            find next chead n2 from GRITn1;
            forward the packet along the local path to n2;
        else
            forward the packet along the local path to n1;
        endif;
    else
        send the packet to its chead n0;
    endif;
else // src = chead
    find next chead n1 from GRITsrc;
    forward the packet along the local path to n1;
endif;

At node x that receives a packet that goes to chead n0
if (x ≠ dst) then
    find next node k along the local path to ch n0;
    if next node k is a chead then
        find next chead n1 w.r.t dst's chead from GRITk;
        if there exists a local path from x to n1 then
            forward the packet along the local path to n1;
        else
            forward the packet to k;
        endif;
    else
        forward the packet to node k;
    endif;
else
    process the packet;
endif;
    
```

그림 3. 적응적 라우팅 알고리즘

그림 2에서 노드 16이 노드 19에게 패킷을 전송하는 경우를 보자. 노드 16은 클러스터헤드 13에게 패킷을 전송하기 전에 GRIT<sub>13</sub>을 참조하여 노드 19가 클러스터헤드 18의 멤버임을 알 수 있다. 또한 클러스터헤드 9가 클러스터헤드 18로 가기 위한 다음 클러스터헤드임을 알게 된다. 노드 16은 9까지 지역 경로 (16, 15, 9)를 알고 있으므로 클러스터헤드 13을 우회하여 노드 15로 패킷을 전송한다. 수신한 노드 15는 다음에 전달할 노드 9가 클러스터헤드이므로 GRIT<sub>9</sub>를 이용하여 다음 클러스터헤드 18을 얻는다. 노드 15는 역시 클러스터헤드 18에 대하여 지역경로 (15, 5, 19, 18)를 알고 있다. 따라서, 클러스터헤드 9를 우회하여 패킷을 노드 5로 전달하고, 계속해서 목적지 노드 19에게 전달된다.

c-DSDV의 경로는 (16, 13, 15, 9, 5, 19)가 되지만, 적응적 라우팅 방식의 경우에 두 클러스터헤드 13, 9를 우회하는 (16, 15, 5, 19)이 경로가 된다.

#### IV. 성능 평가

성능 평가를 위하여 GlomoSim 2.03을 사용하였으며 파라메타 값은 표 4와 같다.

그림 4는 노드 이동성의 변화에 따라 클러스터헤

표 4. 시뮬레이션 파라메타

파라메타	값
노드 이동 패턴	Random Waypoint
노드 이동 속도	5 (m/sec.)
노드 수	100
디멘션 크기	1500 x 3000 (m <sup>2</sup> )
전송 범위	250m
대역폭	2Mbps
트래픽 타입	CBR
CBR 세션 수	10 (전송간격: 2sec.)
데이터 크기	512bytes
시뮬레이션 시간	300(sec.)

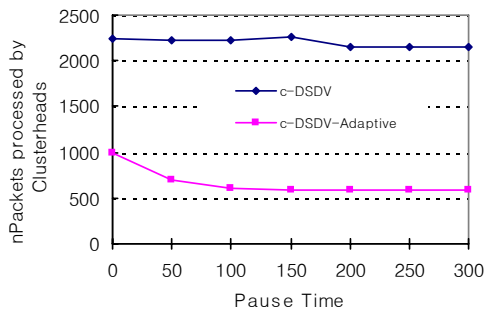


그림 4. 클러스터헤드를 경유하는 데이터 패킷의 수

드의 패킷 처리 오버헤드를 도식한 것이다. X 좌표는 노드가 한 번의 연속적인 이동 후에 쉬는 시간을 나타낸다. Y 좌표는 노드들이 클러스터헤드 상태에 있을 때 전송한 패킷의 수를 모두 합한 값을 나타낸다. c-DSDV-Adaptive의 경우에 c-DSDV에 비해서 패킷 송수신만을 고려하면 클러스터헤드가 처리하는 부하가 약 73% 줄었다 (약 2250개 => 약 600개). 대부분의 전력 소모가 송수신에서 발생한다는 것을 고려하면 이것은 중요한 의미가 있다.

그림 5는 c-DSDV-Adaptive의 경우에 c-DSDV에 비해서 얼마나 양극단 지연시간을 단축할 수 있는지를 보여 준다. 양극단 지연 시간이 늘어나는 원인은 패킷 혼잡으로 인해 큐 길이가 증가하거나 경로 설정 방식의 차이로 인하여 경로 길이가 길어지기 때문이다. c-DSDV의 경우에 어떤 클러스터의 멤버를 지나가는 경로는 반드시 그 멤버의 클러스터헤드를 통과하도록 되어있기 때문에 경로가 최적화되어 있지 않으며, 클러스터헤드를 통과하는 패킷이 많음으로써 클러스터헤드의 큐가 늘어난다. 하지만, c-DSDV-Adaptive의 경우에는 가능한 경우에 클러스터헤드를 우회하도록 하여 클러스터헤드의 혼잡을 피하고 경로 길이를 단축함으로써 40% 이상의 지연시간을 개선 (0.07 -> 0.04)할 수 있었다.

그림 6을 보면, 적응적 라우팅을 통하여 클러스터헤드를 우회하는 패킷들도 여전히 클러스터헤드와

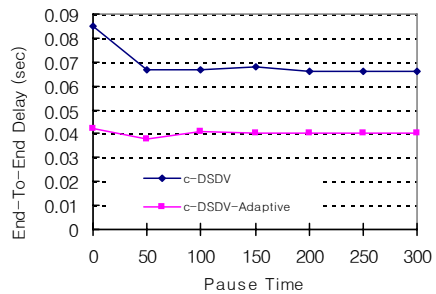


그림 5. 양극단 평균 지연시간

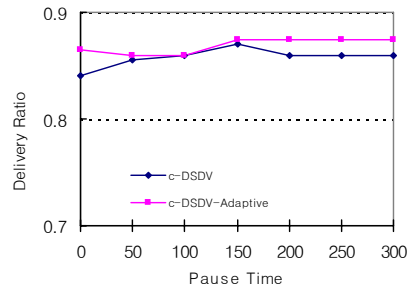


그림 6. 평균 패킷 전송율

중첩되는 무선 통신 대역을 사용하기 때문에 전송율의 개선은 미미하다. 하지만, 패킷 전송 경로 단축으로 인하여 패킷 재전송의 수가 줄어들어 약간의 향상을 보였다. 노드 이동성이 높은 경우에는 콘트롤 메시지의 수가 약간 늘어나기 때문에 c-DSDV의 경우에 더 많이 감소한다는 것을 알 수 있다. 전반적으로 차이가 미미한 다른 이유는 c-DSDV 프로토콜의 콘트롤 오버헤드가 DSDV와 같이 노드 이동성에 대하여 민감하지 않기 때문이다.

## V. 결론

c-DSDV 프로토콜의 클러스터헤드 혼잡 문제를 해결하기 위하여 데이터 패킷을 전송하는 중에 클러스터헤드를 우회할 수 있는지를 판단할 수 있는 방법을 제안하였다. 결과적으로 클러스터헤드 혼잡 문제가 개선되고 경로 길이의 단축으로 종단간 지연 시간이 40% 이상 개선되었다. 제안한 적응적 라우팅 방식은 c-DSDV에 비해 부가적인 통신 대역의 낭비를 전혀 초래하지 않는다.

## 참고 문헌

- [1] M. Gerla and C. Chiang, "Multicluster, Mobile, Multimedia Radio Network," *ACM-Baltzer Journal of Wireless Networks*, vol. 1, no. 3, pp. 255-265, 1995
- [2] M. Jiang, J. Li, and Y. C. Tay, Cluster based routing protocol, July 1999, *Internet Draft. draft-ietf-manet-cbrp-spec-01.txt*.
- [3] 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.
- [4] H. Oh and H. S. Park, "Communication architecture and protocols for broadcast-type mobile multimedia Ad Hoc networks," *2002 Military Communications Conference*, Oct. 2002, pp. 1-6.
- [5] H. Oh, "애드 혹 네트워크에서 클러스터 기반의 DSDV 라우팅 프로토콜(Cluster-Based DSDV Routing Protocol in Ad hoc networks)," *정보통신학회 논문지 '07-6*, vol. 33, 2007
- [6] 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.

- [7] A. Rangaswamy and H. K. Pung, Enhancement of Passive Cluster Based Routing Protocol for Mobile Adhoc Networks, *Proceedings of Eleventh International Conference on Computer Communications and Networks*, pp.376- 381, Oct. 2002

오 훈 (Hoon Oh)

정회원



1981년 성균관대학교 전자공학 학사  
 1993년 텍사스 A&M 대학교 전 산학 석사  
 1995년 텍사스 A&M 대학교 전 산학 박사  
 1996년 삼성전자 중앙연구소 수석  
 2005년~현재 울산대학교 컴퓨터정보통신공학부 조교수  
 <관심분야> 실시간 시스템, 임베디드 시스템, 상황인식 컴퓨팅, 이동 애드 혹 및 센서 네트워크 프로토콜

윤 석 열 (Seok-yeol Yun)

정회원



1995년 2월 강원대학교 재료공학과 졸업  
 1997년 8월 강원대학교 재료공학과 석사  
 2007년 8월 강원대학교 컴퓨터 정보통신공학과 박사  
 2008년 3월~현재 울산대학교 제조정보기술연구센터 박사 후 연구원  
 <관심분야> 이동 애드 혹 및 센서 네트워크, 센서프로토콜 응용

부 초우 투완 (Trong Tuan Vu)

준회원



2007년 베트남 CANTHO 대학교 정보공학 학사  
 2007년~현재 울산대학교 컴퓨터공학전공 석사과정  
 <관심분야> 임베디드 시스템, 이동 애드 혹 및 센서 네트워크