클러스터 기반의 Ad Hoc 네트워크에서 클러스터헤드간 효율적인 통신을 위한 DSP 알고리즘

준회원 윤 석 열*, 정회원 오 훈**°

DSP Algorithm for Efficient Communication between Clusterheads in Cluster-based Ad hoc Networks

Seok-yeol Yun* *Assocate Member*, Hoon Oh**° *Regular Member*

요 약

Ad Hoc 네트워크를 연구한 많은 논문들은 확장성을 향상시키기 위해서 계층적 네트워크 구조를 사용하였다. 계층적 구조는 여러 개의 클러스터들로 구성되고 각 클러스터는 모든 클러스터들에 대하여 정보를 관리하는 클러스터헤드를 가진다. 클러스터헤드는 정확한 정보를 유지하기 위하여 그들끼리 정보를 교환할 필요가 있으며 이를 위하여 클러스터헤드가 이웃하는 다른 클러스터헤드들에게 정보를 보낼 수 있는 효율적인 메커니즘이 필요하다. 이 문제를 해결하기 위한 알고리즘들은 대부분 클러스터헤드가 재전송할 브릿지들을 선정하거나 혹은 부가적인 메시지를 사용하여 재전송 브릿지를 선정하는 메커니즘을 사용하였다. 여기에 제안하는 DSP(Distributed Self-Pruning)알고리즘은 각 노드가 수신한 메시지를 재전송할 것인지에 대하여 독자적으로 판단을 한다. 두 개의 클러스터기반 라우팅 프로토콜에 제안한 알고리즘을 적용함으로써 알고리즘의 적합성을 검증하였다.

Key Words : Ad hoc network, Self-pruning, Control overhead, Cluster

ABSTRACT

Numerous papers that study ad hoc networks have used a hierarchical network structure to enhance scalability. The hierarchical structure typically consists of a number of clusters, each of which has its own clusterhead that maintains information. Clusterheads often need to exchange information among themselves in order to maintain information, and for such cases, a mechanism is needed to efficiently deliver information from one clusterhead to another. Here, we proposed a new distributed algorithm in which every node independently makes the decision about whether or not it forwards a received message. We used a simulation to demonstrate that the algorithm developed for this study is a considerable improvement over the control overhead algorithm..

Ⅰ. 서 론

여러 개의 클러스터들로 구성되고 노드들이 이동을 하더라도 유지 보수되는 계층적인 구조의 Ad Hoc 네트워크가 있다고 가정할 때, 클러스터를 관리하는 클러스터헤드가 이웃하는 모든 클러스터헤드들에게 메시지를 전송해야 하는 경우가 흔히 존재한다. 이 문제를 해결하는 가장 원시적인 방법 중의 하나는 플러딩이며, 모든 노드는 수신한 메시지가 처음 도착한 경우에 재전송을 하는 것이다. 이 경우

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음.
* 강원대학교 컴퓨터정보통신공학과 컴퓨터네트워크 연구실 (kw477@hanmail.net)
** 울산대학교 컴퓨터정보통신공학부 Ubicom 연구실 (hoonoh@ulsan.ac.kr) (° : 교신저자)
논문번호 : KICS2007-02-071, 접수일자 : 2007년 2월 16일, 최종논문접수일자 : 2007년 3월 30일

많은 노드들이 불필요하게 재전송을 함으로써 수신 클러스터헤드는 서로 다른 경로를 따라서 동일한 메시지를 여러 번 받게 됨으로써 네트워크의 효율성을 저하시킨다.

네트워크 브로드캐스트의 효율성을 증가시키기 위한 많은 연구가 있었다[1,2,3,7,9,10]. 이러한 연구는 제안한 라우팅 프로토콜들의 일부분으로써[2,3] 혹은 독립된 브로드캐스트 프로토콜들로서 수행되었다[1,7,8,10]. 주요 프로토콜들의 장.단점을 소개한다.

자기삭제 프로토콜인 SBA[7]에서는 각 노드는 자신의 1홉 거리의 모든 이웃 노드들로 구성된 집합을 브로드캐스트 메시지 (이후부터 BMSG 라고 표기)에 붙여서 전송한다. 수신 노드는 전달한 노드가 커버하지 못하는 적어도 하나의 이웃 노드를 가지고 있다면 재전송을 수행한다. 하지만, 수신 노드가 전달한 노드의 반대편에 노드를 가지는 경우가 허다하기 때문에 그 효과가 아주 제한적이다.

OLSR[2]에서는 모든 노드는 이웃 노드들과 Hello 메시지를 교환함으로써 2홉 거리에 있는 노드들의 정보를 가지며, 모든 이웃 노드들의 특정 부분집합에 속하는 노드들의 전송범위를 합하면 2홉 거리에 있는 모든 노드들을 커버할 수 있는 방식으로 이웃 노드들의 최소 부분 집합인 MPR 셋을 구한다. 그리고 어떤 노드가 메시지를 전송할 때 이 MPR 셋에 속하는 노드들만 재전송에 참여하게 함으로써 재전송의 횟수를 크게 줄인다. 이 방식에서 최적의 MPR 셋을 구하기 위한 알고리즘은 NP-Complete로 알려져 있다. 이 문제를 해결하기 위하여 휴리스틱 알고리즘을 제시하고 있지만 비교적 밀집된 네트워크에서 과도한 연산을 필요로 한다.

이웃 클러스터들의 클러스터헤드들끼리 효율적인 메시지 교환을 위하여 FNSB[10] 알고리즘이 제안되었다 (여기서부터 이웃하는 클러스터들의 클러스터헤드들을 *이웃 클러스터헤드*라고 부른다). 모든 클러스터헤드는 모든 이웃 클러스터헤드들을 커버할 수 있는 *Forward Set* 이라고 불리는 브릿지들의 최소 부분집합을 선정한다. 클러스터헤드는 이렇게 구한 Forward Set을 이웃 브릿지들에게 별도로 전송된다. 결과로, Forward Set에 속한 노드들만 재전송을 수행한다. FNSB는 OLSR 방식과 유사하지만, FNSB는 이웃 클러스터헤드들의 수가 많지 않기 때문에 계산상의 오버헤드는 적다. 클러스터헤드가 재전송할 브릿지를 선정하고 브릿지들에게 미리 알려주어야 한다는 문제가 있다.

NDC[1]에서는 경로 탐색을 시작하기 전에 클러스터를 형성한다. 경로 탐색은 단순한 플러딩이 아니라 클러스터를 이용한 통제된 플러딩을 사용함으로써 수행된다. 클러스터가 형성되기 전에는 초기 노드들, 클러스터가 형성된 후에는 클러스터헤드 및 선택된 브릿지들만이 재전송에 참여한다. 브릿지는 더 많은 이웃 클러스터헤드들을 연결하면 할수록 재전송 브릿지로 선정될 확률이 높다. 동일한 경우에는 클러스터헤드가 아닌 노드들을 더 많이 연결하는 브릿지가 우선권을 갖는다. 문제점은 역시 클러스터헤드가 재전송 브릿지들을 선택해야 한다.

PC[3]에서는 재전송 브릿지가 되기 위하여 별도의 선언 메시지를 사용한다. 각 브릿지는 이전에 선언 메시지를 받지 않았다면 선언 메시지를 보냄으로써 자신이 연결하는 두 개의 클러스터헤드를 선언한다. 이 메시지를 수신하는 브릿지는 자신이 송신 브릿지와 다른 두 클러스터헤드를 연결하고 있지 않으면 재전송 기능을 포기한다. 브릿지들은 Forward Set에 속하는지를 알기위해서는 부가적인 메시지 교환을 해야 하며, 각 브릿지는 한 쌍 이상의 클러스터헤드들을 커버하지 못한다.

본 논문에서 부가적인 메시지를 사용하지 않고, BMSG를 받는 모든 브릿지는 전달 메시지에 부착되어 있는 작은 정보를 이용하여 스스로 재전송을 할 것인지 아니할 것인지를 판단한다. 따라서 이 알고리즘을 실행하기 위해서 필요한 부가적인 오버헤드는 무시할 만하다. 재전송 판단의 계산 복잡도는 n이 이웃클러스터헤드의 수라고 할 때 $O(n)$이다.

2장에서는 문제를 정의하고 해결책을 제시한다. 3장에서 제안한 알고리즘에 대한 자세한 논의를 하고 두 개의 클러스터 기반의 알고리즘, 즉 PCDV[5] 및 GDSR[4] 에 적용함으로써 그 적합성을 판단한다. 4장에서는 결과를 요약한다.

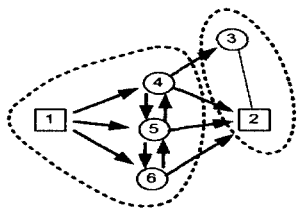


그림 1. 메시지 재전송 예

Ⅱ. 배 경

2.1 문제 정의

클러스터로 형성된 Ad hoc 네트워크에서는 클러스터헤드가 이웃하는 클러스터헤드들에게 메시지를

보낸다고 하자. 메시지는 중간 브릿지들을 경유해서 송신자의 이웃 클러스터헤드들에게 전달된다. 이웃 클러스터들 사이에 복수의 경로를 갖는 경우에는 복수개의 동일한 메시지가 서로 다른 경로를 따라 이웃 클러스터헤드들에게 전달되는 문제점이 있다. 그림 1에 있는 두 개의 클러스터로 구성된 무선 Ad hoc 네트워크를 보자. 두 개의 클러스터헤드(1, 2)는 네 개의 브릿지 (3, 4, 5, 6) 에 의해서 서로 연결되어 있다. 여기서 클러스터 방법은 범위를 벗어나기 때문에 논의를 하지 않는다. 통제된 플러딩을 사용할 때 클러스터헤드 1이 전송하는 메시지가 이웃 클러스터헤드들 (여기서는 2번 뿐임)에게 어떻게 전달되는지를 보자. 먼저 브릿지 4, 5, 6이 그 메시지를 받게 되고, 모두가 처음으로 받았기 때문에 재전송을 수행한다. 다음은 브릿지 3이 같은 식으로 재전송을 한다. 결과적으로 클러스터헤드 2는 4개의 동일한 메시지를 받게 된다.

하나의 방법은 만일 어떤 브릿지가 메시지를 받았을 때 이미 이웃하는 브릿지가 자신이 보내고자 하는 모든 클러스터헤드들에게 메시지를 보냈는지를 알 수만 있다면 재전송을 포기하면 된다. 예를 들면, 브릿지 5가 메시지를 받고 가장 먼저 재전송을 한다고 하자. 브릿지 5가 2를 커버했기 때문에 브릿지 4와 6는 재전송을 포기한다.

제안한 알고리즘의 유효성을 검증하기 위하여 적용할 테이블 기반 라우팅 프로토콜인 PCDV과 요구 기반 프로토콜인 GDSR을 간략히 설명한다.

2.2 PCDV 프로토콜

PCDV는 클러스터링 기법을 이용하는 벡터 테이블 방식의 프로토콜이다. 각 클러스터에는 멤버들을 관리하는 클러스터헤드가 존재하고 클러스터헤드들끼리 하나의 상위 계층의 네트워크 백본을 형성하게 되는데 네트워크 백본에 수정된 DSDV 프로토콜을 적용하였다. 클러스터헤드들은 갱신 메시지를 이웃 클러스터헤드들과 교환함으로써 전역 라우팅 테이블을 구성한다. 전역 라우팅 테이블은 목적지 클러스터헤드로 가기 위한 최단거리에 있는 다음 클러스터헤드를 알려준다. 전역 라우팅 테이블의 각 클러스터헤드에 대한 엔트리는 그 클러스터헤드가 관리하는 멤버 노드들을 포함하기 때문에 목적지 노드가 속한 클러스터헤드를 쉽게 검색할 수 있다. 또한, 토폴로지 변경에 대한 갱신 메시지가 플러딩을 통해서 이웃 클러스터헤드들에게 전달되는 과정에서 이웃 클러스터헤드들 사이에 다중 지역경로가 부수적으로 설정된다. 이렇게 경로가 설정되면 송신 노드는 먼저 자신의 클러스터헤드에게 전송하고 클러스터헤드는 자신의 전역 라우팅 테이블을 이용하여 목적지 노드가 속한 목적지 클러스터헤드를 찾고, 그 다음 목적지 클러스터헤드로 가기 위한 다음 클러스터헤드를 얻는다. 그리고 지역 경로를 이용하여 다음 클러스터헤드로 메시지를 전송한다. 수신한 다음 클러스터헤드는 메시지가 목적지 클러스터헤드에 도달할 때까지 위의 과정을 반복한다. 자세한 설명은 논문[5]를 참조하기 바란다.

2.3 GDSR 프로토콜

GDSR에서는 각 클러스터 멤버에게 고유의 클러스터 레이블을 할당하여 백본을 구성하는 레이블 기반 백본 알고리즘[11]을 사용한다. 이러한 클러스터 백본이 유지되는 환경에서 상대방과 통신을 원하는 노드는 경로를 설정하기 위하여 먼저 경로 탐색 요청 메시지 (RREQ)를 플러딩 한다. 이 때 경로의 구성요소는 노드 번호가 되는 것이 아니고 클러스터를 대표하는 클러스터 레이블이 된다. 따라서 RREQ를 수신한 노드는 RREQ가 동반하는 레이블 경로 목록에 자신의 레이블이 존재하지 않는 경우에만 자신의 레이블을 첨부한다. 수신한 목적지 노드는 경로 탐색 응답 메시지 (RREP)를 구성하여 소스 노드에게 보낸다. RREP는 RREQ을 통해서 설정된 클러스터 레이블 경로의 역경로를 따라 송신 노드로 이동한다. RREQ 메시지가 플러딩 될 때 DSP 알고리즘을 적용할 수 있다. GDSR에 대한 자세한 설명은 논문[4]를 참조하기 바란다.

Ⅲ. DSP 알고리즘

C(b)를 브릿지 b가 메시지를 전송함으로써 직접적으로 (1홉) 혹은 간접적으로 (2홉) 커버할 수 있는 클러스터헤드들의 집합이라고 하자. N(b)를 b의 이웃 브릿지들의 집합이라고 하자. N(b-)를 N(b)에 속하지만 브릿지 b에 앞서 그 메시지를 전송한 브릿지들의 집합이라고 하자. 브릿지 b는 다음 조건이 만족하면 메시지를 그 메시지를 보낼 필요가 없다.

$$C(b) \subseteq \{x \mid x \in C(v),\ v \in N(b-1)\} \qquad (1)$$

브릿지 b가 직.간접적으로 커버할 수 있는 클러스터헤드들에 대하여 커버할 수 있는 클러스터헤드들의 집합인 C(b)를 관리할 수 있다면, 메시지의 재

```
// BMSGi = (msg, coveredSet).
// BMSGi.msg: a message initiated by clusterhead i.
// BMSGi.coveredSet: a set of clusterheads to which BMSGi // has been delivered directly before or at the same time the // message arrives at a certain bridge.
// coverableSeti : the set of bridge i's neighbor clusterheads // and the clusterheads of bridge i's neighbor bridges that
// belong to bridge another cluster.
⊙ sending clusterhead i:
    send BMSGi = (msg, coveredSet);
⊙ clusterhead j that receives BMSGi:
    Process BMSGi;
    free(BMSGi);
⊙ ordinary node l that receives BMSGi:
    free(BMSGi);
⊙ bridge k that receives a BMSGi
    S=coverableSetk - {(src,d)|d=1 or 2};
// α is delay coefficient and randomDelay is a random delay
    delay-jitter = α * (|S| - |S|d-1) * randomDelay;
    set BMSGi's transmission delay to delay-jitter;
    put the BMSGi into the queue and execute
    the timer;
⊙ bridge k whose timer expires
// S includes all covered sets received along different routes
  for each coveredSet v in the queue that belongs to BMSGi do
    S = S ∪ v;
  endfor;
// i is the source node that initiated BMSG
  S=coverableSetk - {(i, d)} - {(x,1), (x,2)| (x,d) ∈ S};
  if (S != ∅) then
    S' = {(x, d)| (x, d) ∈ S, d = 1};
    resend BMSGi = (msg, S');
  elseif;
  delete all queued messages with respect to BMSGi;
```

그림 2. DSP 알고리즘

전송에 관한 효율적인 판단을 할 수 있게 된다. 다행히도, 부가적인 컨트롤 메시지를 사용함이 없이 Hello 메시지만을 사용하여 각 브릿지가 커버할 수 있는 클러스터헤드들의 집합을 관리할 수 있다. Hello 메시지를 이용하여 클러스터구조에 대한 정보의 교환, 링크 연결성의 감지, 그리고 커버할 수 있는 집합의 관리 등을 할 수 있다. 결과적으로, 메시지를 수신하는 모든 노드는 다음 두 가지 조건이 동시에 만족되는 경우 재전송을 하게 된다.

(1) 수신 노드가 브릿지이다. 그리고
(2) Not C(b) ⊆ {x| x ∈ C(v), v ∈ N(b-1)}.

클러스터헤드 i에 의해 생성되는 브로드캐스트 메시지를 $BMSG_i$ 라고 하자. 편의상 두 개의 표기, 즉 $coverableSet_i$, $BMSG_i.coveredSet$를 사용한다. $coverableSet_i$는 브릿지 i의 이웃 클러스터헤드들과 다른 클러스터에 속하는 브릿지 i의 이웃 브릿지의 클러스터헤드들의 집합을 나타낸다. 집합의 요소는 클러스터헤드 ID와 그 클러스터헤드까지의 거리로 구성된다. 그림 3에서, $coverableSet_{10}$={(1,1),(2,2)}. 집합에 속하는 모든 클러스터헤드는 최대 2홉이며, 따라서 그러한 정보는 Hello 메시지를 사용하여 쉽게 얻을 수 있다. $BMSG_i.coveredSet$ 는 해당 메시지가 어떤 브릿지에 도착하기 전, 또는 동시에 직접 전달되었던 클러스터헤드들의 집합이다. 모든 브릿지는 수신된 메시지를 전달할 때 coveredSet을 동반한다. 결과로써 모든 수신 브릿지들은 어떤 이웃 클러스터헤드들이 해당 메시지를 이미 수신하였는지를 알 수 있다. DSP (Distributed Self-Pruning) 알고리즘은 그림 2에 상세히 기술되어 있다.

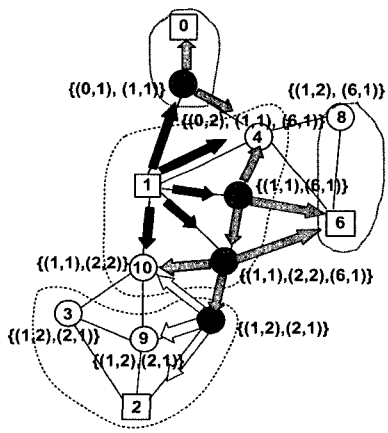


그림 3. 최적화된 메시지 전송

그림 3을 보면, 각 브릿지는 자신의 coverableSet을 관리한다. 즉, 브릿지 13은 클러스터헤드 0과 1을 직접 커버할 수 있다. 하지만, 브릿지 13과 클러스터헤드 6은 2-홉 거리에 있음에도 불구하고 서로 이웃 클러스터에 속하지 아니함으로 브릿지 13은 클러스터헤드 6을 커버할 수 없다. 클러스터헤드 1이 $BMSG_1$을 이웃 클러스터헤드 0, 2, 6으로 보낸다고 하자. 소스 클러스터헤드는 직접 커버할 수 있는 클러스터헤드들이 없기 때문에, coveredSet=∅으로 설정한 후 $BMSG_1$에 붙여서 전송하고, 결과적으로 브릿지 4, 5, 7, 10, 13은 $BMSG_1$을 수신한다. 노드 5가 맨 처음으로 재전송 판단을 한다고 가정

하자. 이 경우 coverableSet$_5$ = {(1,1),(6,1)}로 설정된다. 브릿지 5는 클러스터헤드 1과 6을 동시에 커버하지만, 노드 1은 BMSG의 생성자이기 때문에, coveredSet ={6}을 가진 BMSG를 전송하게 되고 브릿지 4, 7이 그것을 수신한다. 다음은 브릿지 7과 13이 재전송 판단을 한다고 가정하자. 노드 13은 클러스터헤드 0을 커버할 수 있으므로 coveredSet={0}를 포함한 BMSG$_1$ 메시지를 재전송한다. 브릿지 4는 자신의 coverableSet$_4$에 포함된 모든 클러스터헤드들이 두 이웃 브릿지 5와 13에 의해 이미 커버되었음을 알기 때문에 재전송을 포기한다. 클러스터헤드 6을 직접 커버할 수 있는 브릿지 7은 coveredSet={6}을 포함한 BMSG$_1$을 재전송한다. 수신 브릿지 10과 11중에서, 브릿지 11이 먼저 재전송 판단을 한다고 가정하자. 브릿지 11은 coveredSet={2}를 포함하는 BMSG$_1$을 재전송한다. 수신 브릿지 10은 클러스터헤드 2가 이미 브릿지 11에 의해 커버되었음을 알게 된다. 예에서 약 50%의 브릿지가 재전송을 포기한다는 것을 알 수 있다.

Ⅳ. 성능 분석

Glomosim 2.03시뮬레이터[8]을 사용하여 DSP 알고리즘을 PCDV와 GDSR에 적용하였으며, 표 1에 제시한 파라메타 값들을 사용하였다.

표 1. 시뮬레이션 파라메타

파라메타	값
노드 이동 패턴	Random Waypoint
노드 최대 이동속도	5 (m/sec), 15 (m/sec)
정지 시간	30 (sec)
노드 수	50, 75, 100, 150, 200, 250
디멘젼 크기	1500 x 300
전송 거리	250 (m)
대역폭	2 (Mbps)
트래픽 유형	CBR (세션 : 10 개)
시뮬레이션 시간	300 (sec)

그림 4와 5는 각 이동속도 5 m/sec와 15 m/sec에서 노드 수의 변화에 대한 컨트롤 오버헤드의 변화를 나타낸다. 그래프 범례의 X-Y에서 X는 프로토콜 명, Y는 이동속도를 나타낸다. 사용된 프로토콜은 PCDV, GDSR이며, DSP 알고리즘이 적용된 경우에 프로토콜 명은 PCDV-DSP, GDSR-DSP이다. PCDV-DSP-5는 PCDV-5에 비해 밀도에 관계없이 50%이상의 오버헤드 감소를 보였다. 그림 5에서 PCDV-15의 경우에는 이동속도 15m/sec에서 노드의 빈번한 이동으로 인하여 BMSG에 포함되어 전달되는 멤버의 수가 증가하기 때문에 노드 수가 200 이상에서 컨트롤 오버헤드가 급격히 증가한다. 하지만, PCDV-DSP-15의 경우에는 여전히 완만한 증가를 보인다.

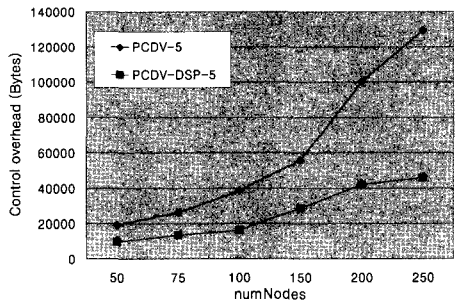


그림 4. PCDV와 PCDV-DSP의 컨트롤 오버헤드 비교

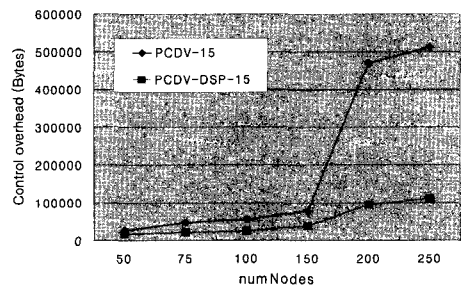


그림 5. PCDV와 PCDV-DSP의 컨트롤 오버헤드 비교

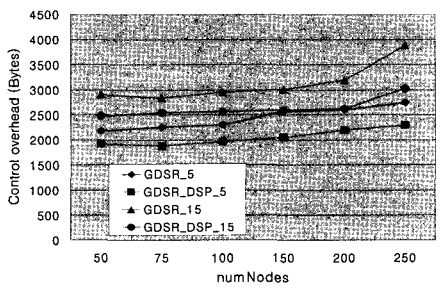


그림 6. GDSR과 GDSR-DSP의 컨트롤 오버헤드 비교

그림 6은 DSP 알고리즘을 요구기반 프로토콜인 GDSR에서는 밀도가 높아지는 경우에 그룹 사이에 연결성이 증가하고, 이로 인하여 경로 파손 시 대부분 지역 복구가 가능하기 때문에 RREQ의 재송신 빈도가 증가하지 않는다. 따라서 노드 밀도가 증가

함에도 불구하고 GDSR-DSP의 오버헤드는 크게 증가하지 않는다. 속도의 증가에 대하여도 큰 차이가 없음을 알 수 있다.

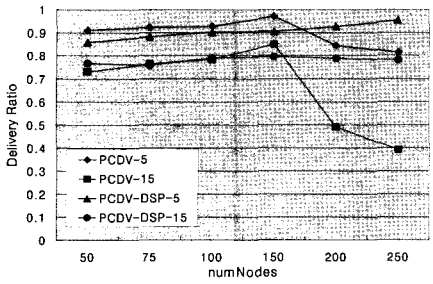


그림 7. PCDV와 PCDV-DSP의 전송률 비교

그림 7은 전송률을 도시한 것이다. PCDV-5와 PCDV-15는 노드 수 150 이하에서는 전송률이 완만하게 증가하나 노드 수 150 이상에서는 데이터 패킷과 컨트롤 오버헤드의 증가로 인하여 전송률이 급격히 감소한다. 그러나 PCDV-DSP 프로토콜의 경우에는 DSP의 영향으로 노드 수가 증가함에도 일정한 전송률을 보이고 있다. 이는 DSP의 적용으로 네트워크 확장성이 크게 증가하였음을 보여준다.

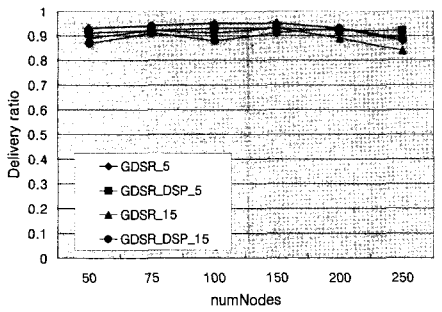


그림 8. GDSR과 GDSR-DSP의 전송률 비교

그림 8은 GDSR과 GDSR-DSP의 전송률을 도시한 것이다. GDSR 프로토콜에서는 노드 수가 많아지면 인접 클러스터끼리 연결성이 증가하고 지역 복구 수가 증가하여 RREQ의 영향을 덜 받는다. 따라서 RREQ의 재전송에 DSP 알고리즘을 적용하더라도 그 영향이 커지 않음을 알 수 있다. 그럼에도 불구하고, 노드 수가 200 이상인 경우에 약간의 영향이 있음을 알 수 있다.

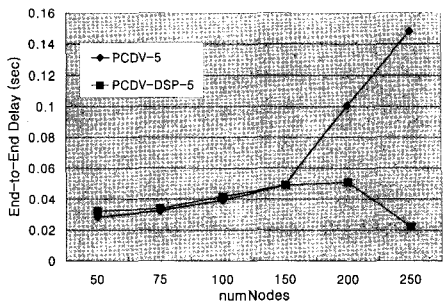


그림 9. 종단간 지연시간 비교 (이동속도: 5m/sec)

그림 9는 종단간 지연시간을 비교한 것이다. 노드 수 150 이상에서 PCDV는 종단간 지연시간이 급격히 증가하나 PCDV-DSP-5는 오히려 감소한다. 이는 노드 수의 증가에 따른 오버헤드의 변화와 일치하는 현상이다. PCDV-DSP-5의 경우에 노드 수의 증가와 더불어 오히려 안정성이 증가하는 현상은 이웃 클러스터헤드들 간에 연결성의 증가로 인한 경로의 안정성이 증가하기 때문이다.

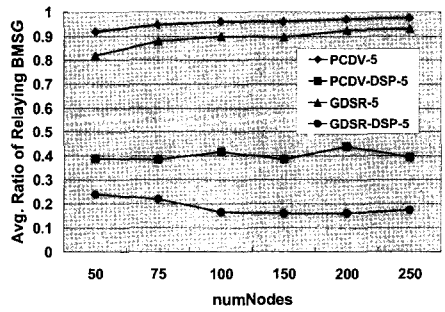


그림 10. BMSG의 평균 재전송률

DSP 알고리즘을 사용하는 경우 모든 컨트롤 메시지는 coveredSet을 동반하기 때문에 크기가 약간 커진다. 브로드캐스트 메시지는 CSMA/CA 프로토콜에 적용을 받지 않기 때문에 패킷 충돌의 주요 원인이 된다. 따라서 큰 크기의 브로드캐스트 메시지를 작은 크기의 브로드캐스트 메시지로 쪼개서 여러 번 전송하는 것이 훨씬 더 네트워크 성능에 악영향을 끼치게 된다. 그림 10은 DSP알고리즘을 적용하는 경우에 브로드캐스트되는 컨트롤 메시지의 수가 얼마나 감소하는지를 보여주는 그래프이다. 적용된 프로토콜에 따라서 재전송률이 60%에서 80%까지 향상 되었다.

V. 결 론

재전송에 참여하는 브릿지의 수를 줄이기 위해 제안한 DSP 알고리즘은 기존의 다른 방식들과는 달리 재전송 노드를 미리 선정하지 않는다. 각 브릿지는 수신한 메시지에 포함된 작은 정보를 검사하여 스스로 재전송 판단을 수행한다. 노드 밀도 와 데이터 트래픽이 많을수록 DSP 알고리즘의 적용 효과가 높아짐을 알 수 있었다.

참 고 문 헌

[1] T.-C. Chiiang, P.-Y. Wu, Yueh-Min Huang, "A limited flooding schmeme for mobile ad hoc networks," *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference*, vol. 3, pp. 473-478, Aug. 2005.

[2] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot, "Optimized link state routing protocol for ad hoc networks." *In IEEE International Multi Topic Conference*, 2001.

[3] Taek Jin Kwon, Mario Gerla, "Efficient flooding with passive clustering(PC) in ad hoc networks," *ACM Computer Communication Review*, vol. 32, no. 1, pp. 44-56, Jan. 2002.

[4] D. M. Ngoc, H. Oh, "A Group Dynamic Source Routing Protocol for Ad Hoc Networks," *Proc. 1st IFOST 2006*, pp. 134-137, Oct. 2006.

[5] H Oh, S.Y. Yun, "Proactive cluster-based distance-vector(PCDV) routing protocol in mobile ad hoc networks", *IEICE Trans. On Communications*, Vol. E90-B, No. 6, Jun. 2007.

[6] H. Oh, H. S. Park, "Communication architecture and protocols for broadcast-type mobile multimedia ad hoc networks" *MILCOM 2002. Proceedings*, vol. 1, 7-10 , pp. 442 – 447, Oct. 2002.

[7] W. Peng, X.-C. Lu, "On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks," *Proc. First Ann. Workshop Mobile and Ad Hoc Networking and Computing*, pp. 129-130, Aug. 2000.

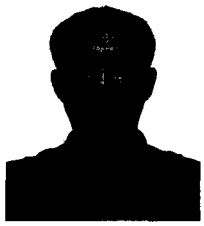
[8] UCLA Parallel Computing Laboratory and Wireless Adaptive Mobility Laboratory. GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems, http://pcl.cs.ucla.edu/projects/glomosim

[9] I. Stojmenovic, M. Seddigh, J. Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14-25, Jan. 2002.

[10] J. Wu, W. Lou, "Forward-node-set-based broadcast in clustered mobile ad hoc networks," *Wireless Communications and Mobile Computing, a special issue on Algorithmic, Geometric, Graph, Combinatorial, and Vector Aspects of Wireless Networks and Mobile Computing*, vol. 3, no. 2, pp. 141-154, March 2003.

[11] Vitaly Li, H. S. Park and H. Oh, "A Cluster-Label Based Mechanism for Backbones on Mobile Ad Hoc Networks, *LNCS*, vo. 3970, pp. 26-36, May 10-12, 2006

윤 석 열 (Seok-yeol Yun) 준회원



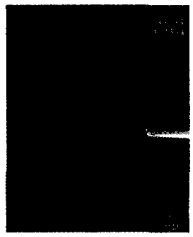
1995년 : 강원대학교 재료공학 학사

1997년 : 강원대학교 재료공학 석사

1999년~현재 : 강원대학교 컴퓨터정보통신공학과 박사수료

<관심분야> 애드 혹 네트워크 프로토콜, 센서 네트워크

오 훈 (Hoon Oh) 정회원



1981년 : 성균관대학교 전자공학 학사

1993년 : 텍사스A&M대학교 전산학 석사

1995년 : 텍사스A&M대학교 전산학 박사

1996년 삼성전자 중앙연구소 수석 연구원

2005년~현재 : 울산대학교 컴퓨터정보통신공학부 조교수

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