무선 이동 애드 혹 네트워크를 위한 동적 그룹 소스 라우팅 프로토콜

준회원 곽 운 용*, 정회원 오 훈*°

Group Dynamic Source Routing Protocol for Wireless Mobile Ad Hoc Networks

Woon Yong Kwak* *Associate Member*, Hoon Oh*° *Regular Member*

요 약

이동 애드 혹 네트워크에서는 노드의 이동성으로 인해 설정된 라우팅 경로를 안정적으로 유지하는 것이 어렵다. 본 논문에서는 사실상의 복수 경로를 의미하는 그룹경로를 사용하여 경로 안정성을 높인 그룹 소스 라우팅 프로토콜을 제안한다. 물리적으로 인접한 노드들끼리 클러스터 (혹은 그룹)를 형성하며, 각 클러스터는 모든 멤버들을 직접 연결하는 클러스터헤드와 클러스터를 유일하게 식별하기 위한 클러스터 레이블을 갖는다. 그룹경로는 소스에서 목적지까지 놓여있는 클러스터들의 레이블 시퀀스로 이루어진다. 그룹경로상의 각 클러스터에 속한 노드들은 클러스터들을 잇는 브릿지 노드들을 통해 그룹경로상의 다음 레이블을 갖는 클러스터의 노드로 패킷을 전달한다. 따라서, 그룹경로는 단일 링크 단절에 의해 쉽게 깨어지지 않고, 경로상의 인접한 그룹 간에 연결이 완전히 끊어지지 않는 한 유효하다. 시뮬레이션을 통해 높은 이동성과 높은 트래픽 상황에서도 그룹 라우팅 프로토콜이 다른 주요 소스 라우팅 프로토콜들보다 안정성이 뛰어나다는 것을 입증하였다.

Key Words : Mobile Ad Hoc Networks, Group Path, Cluster Group, Source Routing, Scalability

ABSTRACT

It is very hard, but important to sustain path stability for a reliable communication in mobile ad hoc networks. We propose a novel source routing protocol that establishes a group path with virtual multiple paths to enable a robust communication. The entire mobile nodes form a disjoint set of clusters: Each has its clusterhead as a cluster leader and a unique cluster label to identify itself from other clusters. A group path is a sequence of cluster labels instead of nodes and the nodes with the same label collaborate to deliver packets to a node with next label on the group path. We prove by resorting to simulation that our proposed protocol outperforms the existing key routing protocols, even for a network with a high node mobility and a high traffic.

I. 서 론

최근 십여 년에 걸쳐서 이동 애드 혹 네트워크(MANET)의 라우팅에 관련된 많은 연구가 진행되었다. MANET는 응용 분야에 따라 노드 이동성, 노드 밀도, 세션의 수, 네트워크 크기가 크게 변하기 때문에 네트워크 전개 시나리오의 확장성에 효과적으로 대응할 수 있는 안정성이 우수한 라우팅

※ 본 연구는 울산대학교 교내 연구비로 수행되었음.
* 울산대학교 컴퓨터정보통신공학부 UbiCOM 연구실 (text_rod@yahoo.co.kr), (hoonoh@ulsan.ac.kr)(° : 교신저자)
논문번호 : KICS2008-01-031, 접수일자 : 2008년 1월 15일, 최종게재논문통보일자 : 2008년 9월 30일

프로토콜의 설계는 여전히 연구의 주요 이슈이다. 본 논문에서는 소스에서 목적지까지 사실상의 복수 경로를 갖는 그룹경로를 설정하는 소스 라우팅 프로토콜을 제안함으로써 이러한 문제를 해결하고자 한다.

라우팅 안정성을 개선하기위한 연구들은 주로 다음 4가지 측면에서 수행되었다. DSDV [7], AODV [8]에서처럼 콘트롤 오버헤드를 낮추는 접근 방식, DSR [3], CBRP [4], PCDV[6]에서처럼 복수 경로 설정을 통한 방식, ZRP [2], CBRP, PCDV에서처럼 계층적인 네트워크 구조를 사용함으로써 경로 설정과 복구의 효율성을 개선하는 방식, PCDV에서처럼 토폴로지 변경에 대한 수렴속도 개선 방식이 있다. DBF [1]를 개선한 DSDV는 점진적 정보 갱신 및 루핑 문제 해결을 통해서 오버헤드를 줄였다. PCDV에서는 각 클러스터헤드들이 모든 클러스터헤드들의 각각에 대하여 하나의 엔트리를 갖는 전역 라우팅 테이블을 구성하고 이웃 클러스터헤드들 사이에 있는 노드들은 지역 라우팅 테이블을 구성하여 테이블의 크기를 줄인다. 따라서, PCDV는 갱신 메시지의 크기 축소, 이웃 클러스터헤드들 사이의 복수 경로 설정, 토폴로지 변경에 대한 수렴 속도 개선 등을 통해서 라우팅의 안정성을 크게 높였다. 노드 이동성이 높으면 (15m/s 이상) 다소 경쟁력이 떨어진다.

한편, 노드 이동성이 높은 이동 애드 혹 네트워크를 위한 여러 가지 요구기반 라우팅 프로토콜들이 제안되었다. 대표적인 프로토콜인 DSR과 AODV는 통신을 위한 경로가 필요한 경우에만 라우트 탐색 메시지를 사용하여 경로를 찾는다. 이 두 방식의 차이점은 전자는 패킷이 탐색된 경로를 가지는 소스 라우팅을 수행하고, 후자는 설정된 경로 상의 각 노드가 목적지 노드로 가는 다음 노드에 대한 포인터를 저장함으로써 라우팅을 수행한다는 것이다. DSR의 경우에 각 노드가 라우트 탐색 메시지, 라우트 응답 메시지 혹은 데이터 패킷을 수신할 때 습득한 경로정보를 자신의 캐시 메모리에 저장함으로써, 라우트 탐색 혹은 복구의 효율성을 높인다. 하지만, DSR은 패킷 헤드의 크기 증가, 경로 상의 단일 링크 파손으로 인한 경로 재탐색, 그리고 캐쉬 관리의 어려움으로 인해 유효하지 않은 경로를 활성 경로로 사용할 수 있는 문제가 있다. AODV의 경우에도 순간적인 오버헤드 증가와 단일 링크 파손으로 인한 경로 재탐색 오버헤드의 문제가 있다.

하이브리드 방식인 ZRP 및 CBRP는 근접 지역에서는 토폴로지 정보 관리를 통해서 라우팅을 수행하고, 근접 지역외의 목적지에 대해서는 요구 기반으로 경로를 설정한다. CBRP는 클러스터헤드들의 도움으로 경로 탐색의 효율성을 높이고, 응답 메시지를 전송하는 동안에 각 노드에서 관리되는 2-홉 토폴로지 정보를 이용하여 경로를 최적화한다. 어떤 노드가 데이터 전송 중에 링크 파손을 탐지하면 소스 노드에 에러 메시지를 보내고 2-홉 토폴로지 정보를 이용하여 자신보다 하위에 있는 어떤 노드에 대한 우회 경로를 찾는다. 우회 경로를 찾으면, 목적지로 가는 해당 패킷을 구제한다. 목적지 노드는 구제 패킷을 받는 즉시 무료 응답 메시지 (gratuitous route reply message)를 소스에 보냄으로써 새로운 경로를 설정한다. CBRP는 경로 탐색을 위하여 클러스터헤드들에 의존하고, 설정된 경로는 단일 링크 파손으로 인하여 사용할 수 없게 된다는 문제점이 있다.

지금까지 논의된 라우팅 프로토콜들에서는 단일 링크 파손이 경로 파손의 결과가 된다. 본 논문에서는 복수의 클러스터들과 각 클러스터 내에 모든 멤버들을 직접 연결하는 핵심 멤버인 클러스터헤드를 갖는 클러스터 네트워크 토폴로지를 구축하고, 각 클러스터에는 클러스터를 유일하게 식별하기 위한 클러스터 레이블이 할당 된다 [5]. 경로 탐색에서는 클러스터 레이블들의 시퀀스로 표현되는 그룹경로를 설정 및 사용함으로써 경로의 안정성을 높인다. 노드 이동으로 인한 토폴로지 변화에도 불구하고 클러스터 멤버들이 공유하는 클러스터 레이블이 보전되는 식으로 클러스터들을 관리함으로써 그룹경로는 유지된다. 패킷 전송 시에는 동일한 레이블을 갖는 노드들이 협력하여 그룹경로 상의 다음 레이블을 가진 클러스터의 멤버로 패킷을 전달한다.

또한, 노드 밀도가 높은 네트워크에서는 클러스터에서 클러스터로 이동하는 경로 탐색 메시지가 서로 다른 브릿지 노드들에 의해서 중복으로 전송될 수 있다. 이 문제를 해결하기 위하여 분산 자기 삭제 (DSP: Distributed Self-Pruning) [9]알고리즘을 적용하였으며, 그 효과를 평가하였다.

이 논문의 구성은 다음과 같다. Ⅱ장은 연구 배경을 설명하고, Ⅲ장은 제안하는 프로토콜을 설명한다. Ⅳ장에서는 수정한 DSP 알고리즘을 설명하고, Ⅴ장에서는 성능 평가, 그리고 Ⅵ장에서 결론을 맺는다.

Ⅱ. 배 경

2.1 문제 정의

DSR은 RREQ (경로 탐색 메시지)를 플러딩하고, 목적지 혹은 목적지까지 경로를 알고 있는 노드로부터 RREP (경로 응답 메시지)를 수신함으로써 경로를 얻고, 모든 전송패킷들은 경로를 동반한다. 전송 중에 경로상의 링크가 끊어지면 경로 재탐색을 위하여 소스 노드로 RERR(에러 메시지)를 보낸다. 부분경로 혹은 경로를 동반하는 RREQ, RREP 혹은 패킷을 수신하는 모든 노드는 이를 통해 습득한 경로를 자신의 캐쉬에 저장한다. RREQ를 수신하는 어떤 중간 노드가 목적지까지 유효경로를 가지고 있다면 목적지를 대신하여 자신이 RREP 메시지를 보낼 수 있다. 캐쉬에 저장된 각 경로에 타이머를 할당하여 경로의 유효성을 관리한다. 타이머가 길면 유효하지 않은 경로들이 캐쉬에 있을 수 있으며, 타이머가 짧으면 실제 유효한 많은 경로들이 삭제될 수 있다. 그림 1-(a)는 노드 2가 패킷 전송 중에 링크 파손을 감지하고 RRER을 소스에게 보냄으로써 경로 재탐색을 요청하는 경우를 보여준다.

그림 1-(b)는 CBRP의 경로 설정 및 복구 과정을 보여준다. 패킷 전송 중에 링크 (2, 5)가 파손되면 이를 감지하는 노드 2는 RERR을 소스 노드에게 보내고 자신의 2-홉 토폴로지 정보로부터 우회하는 경로를 찾는다. 우회 경로 (2, 6, 3)을 통해서 해당 패킷을 구제한다. 목적지 D가 구제된 패킷을 수신하면 소스 노드로 무료로 RREP를 보냄으로써 새로운 경로 (S, 1, 2, 6, 3, D)가 설정된다. RERR을 받은 소스가 일정 시간 내에 무료 RREP를 받지 못하면 경로 복구가 실패한 것으로 판단하고 새로운 경로 탐색을 실시한다. 하지만, 링크 (1, 2)가 파손되는 경우에는 노드 1은 패킷을 구제할 수 없다 (경로상의 노드 5, 3, D 중의 하나도 2홉 내에 있지 않음).

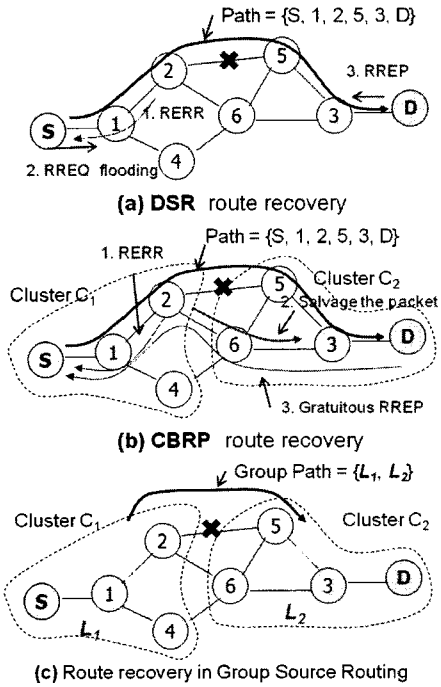


그림 1. 주요 소스 라우팅 프로토콜의 경로 복구 복잡도 비교

2.2 해결 방안

DSR이나 CBRP에서는 경로상의 하나의 링크가 파손되면 경로가 파손된다. 그룹경로를 사용하는 경우에는 패킷을 가진 노드는 먼저 그룹경로상의 다음 레이블을 가진 노드를 찾거나 다음 레이블을 가진 어떤 노드에 대한 연결을 가지고 있는 노드를 찾는다. 이러한 노드가 없는 경우에는 자신의 클러스터헤드에게 전송하고 클러스터헤드는 동일한 절차를 취한다. 그림 1-(c)에서 두 클러스터의 레이블을 각각 L_1, L_2이라고 할 때, 그룹경로는 (L_1, L_2)이다. S는 선택의 여지가 없기 때문에 자신의 클러스터헤드로 패킷을 전송하고, 클러스터헤드 1은 노드 2로 전송한다 (낮은 ID를 먼저 선택한다고 가정). 전송 중에 만일 링크 (2, 5)가 파손되면 노드 2는 이웃 노드 정보를 이용하여 패킷을 노드 6으로 즉시 보낸다. 또한, 링크 (1, 2)가 파손된 경우에도 CBRP와는 달리 노드 1은 브릿지 4를 즉각 선택할 수 있다.

경로상의 인접한 두 클러스터 사이에 연결이 완전히 끊어지지 않는 한 그룹경로는 유효하다는 것을 알 수 있다. 더구나, 패킷 전송 중에 그룹경로 상의 인접한 두 클러스터 사이에 새로운 링크가 추가 되면, 그룹경로의 안정성은 강화된다. 또한, 경로 탐색 과정에서 CBRP와는 달리 클러스터헤드에 의존하지 않으며, 패킷 전송을 할 때마다 노드는 지역적으로 덜 혼잡한 노드들 선택할 수 있다.

그룹경로의 안정성을 확보하기 위해서는 다음과 같은 두 가지 요소가 선행되어야 한다. 노드들의 동적 변화에 대하여 클러스터를 안정적으로 유지하기 위한 클러스터 관리 방법을 제시해야 한다. 또한 이웃 노드들의 연결정보를 신속 정확하게 관리할 수 있는 방법을 제시하여야 한다. 이 두 가지 문제의 해결 방안에 대하여 자세히 논의한다.

2.3 시스템 모델 및 표기

노드들은 서로 충첩되지 않은 클러스터들을 형성

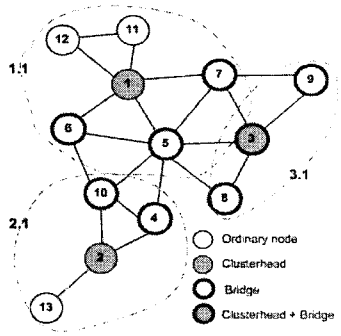


그림 2. 클러스터 구성 및 레이블 할당

하고, 클러스터헤드는 클러스터 내의 모든 다른 노드들을 연결한다. 브릿지는 서로 다른 클러스터를 연결하는 노드이며, 클러스터헤드는 브릿지 역할을 할 수 있다. 일반 (Ordinary) 노드는 클러스터헤드나 브릿지가 아닌 클러스터 멤버를 말하며, 고아 (Orphan) 노드는 클러스터 멤버가 아닌 노드를 말한다. 동일한 클러스터에 속하는 모든 멤버 노드들은 자신의 클러스터헤드가 생성하는 동일한 클러스터 레이블을 갖는다. 클러스터 레이블은 점으로 분리되는 클러스터헤드 주소와 카운터 번호로 구성된다 [5]. 그림 2는 클러스터 레이블 1.1, 2.1 및 3.1이 백본을 형성하고 있는 클러스터 토폴로지를 나타낸다.

클러스터 연결 정보 관리를 위하여 클러스터 연결 목록 (CCL: Connecting Cluster List)과 브릿지 목록 (BL: Bridge List)을 사용한다. 모든 브릿지 i는 자신이 연결하는 클러스터들의 클러스터 레이블을 포함하는 CCL_i를 관리하고, 모든 노드 i는 자신의 이웃 레이블의 각각에 대하여 그 레이블을 가진 클러스터에 직접 연결하는 브릿지들을 포함하는 BL_i을 사용한다. 모든 노드들은 자신의 CCL을 포함하는 Hello 메시지를 주기적으로 전송하고 수신 노드는 자신의 CCL과 BL을 갱신한다.

그림 2에서 10개의 브릿지 즉, 3, 4, 5, 6, 7, 8, 9, 10이 있으며, CCL_3 = {1.1}, CCL_4 = {1.1}, CCL_5 = {2.1, 3.1}, CCL_6 = {2.1}, CCL_7 ={3.1} 등, 그리고 BL_1 = {(2.1 {5, 6}), (3.1, {5, 7})}, BL_2 = {(1.1, {4, 10})}, BL_3 = {(1.1, {5, 7, 8})} 등으로 나타낸다.

Ⅲ. 클러스터 구조 관리

그룹경로의 안정성을 확보하기위해서 네트워크 클러스터 구조의 효율적인 관리는 매우 중요하다.

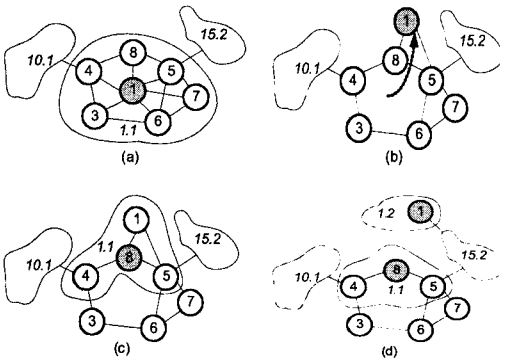


그림 3. 클러스터 유지 및 보수

멤버가 클러스터를 떠나는 경우에 유지 보수는 간단하다. 하지만, 클러스터헤드가 이동하는 경우에는 문제가 복잡해진다. 클러스터 관리의 주요 원칙은 클러스터헤드가 자신의 클러스터를 떠난다고 할지라도 멤버들은 자신의 클러스터 레이블을 보존하도록 함으로써 해당 클러스터을 포함하는 그룹경로가 유지되게 하는 것이다.

클러스터헤드가 이동하면 다른 클러스터의 멤버가 되거나 혹은 스스로 새로운 클러스터를 형성하고 새로운 클러스터헤드가 되기도 한다. 새로운 클러스터헤드가 되는 경우에는 레이블의 유일성을 유지하기위하여 카운터 번호를 1만큼 증가시킨다. 만일, 클러스터헤드가 이동하여 어떤 하나의 이웃 클러스터와 모든 연결이 끊어짐을 감지하는 경우에 자신을 대신할 새로운 클러스터헤드를 선정하기 위하여 메시지를 브로드캐스트 한다. 이를 수신한 멤버 노드들은 자신이 클러스터헤드가 될 수 있는지를 판단하고 응답한다. 이 때 새로운 클러스터헤드 선정은 클러스터 연결구조의 안정성을 높이는 식으로 우선순위가 주어진다. 다음과 같은 선정 기준을 적용한다.

1) 기존의 클러스터헤드가 연결하였던 모든 이웃 클러스터를 연결하는 노드; 그리고

2) 기준 1에 대하여 두 개 이상의 노드가 만족하는 경우에는 다수의 이웃을 갖는 노드를 선정한다.

예를 들면, 노드 1이 그림 3-(b)에서처럼 이동하여 레이블 10.1을 가진 클러스터를 연결하고 있는 브릿지 4와 연결이 끊어지는 경우를 보자. 이를 탐지한 클러스터헤드 1은 UpdateState 메시지를 브로드케스트한다. 이 메시지를 받는 노드들 중에서, 선정 기준 1에 따라서, 그림 3-(c)에서 처럼 노드 5 대신에 노드 8이 클러스터헤드가 된다. 노드 1은

새로운 클러스터헤드 8로부터 UpdateState 메시지를 받으면 멤버가 된다. 노드 1이 계속 움직여서 그림 3-(d)에서처럼 새로운 클러스터헤드 8로부터 연결이 끊어진다면 그 노드는 새로운 클러스터를 형성하고, 클러스터레이블의 유일성을 보장하기 위하여 새로운 레이블의 카운터는 1만큼 증가하고 레이블은 1.2가 된다. 만일 그룹경로가 (10.1, 1.1, 15.2)이라면 클러스터헤드의 움직임 후에도 그 경로는 보존된다.

Ⅳ. 동적 그룹 소스 라우팅

이 장에서는 동적 그룹 소스 라우팅 (GDSR: Group Dynamic Source Routing)의 그룹경로 탐색, 패킷 전송 및 복구 방식에 대하여 자세히 설명한다.

4.1 그룹경로 탐색

패킷 전송을 원하는 노드는 RREQ = (destAddr, srcAddr, seqNum, *ll*)를 사용하여 경로 탐색을 시작한다. destAddr은 목적지 노드의 주소, srcAddr은 소스 노드의 주소, seqNum는 새로운 RREQ를 보낼 때 마다 증가하는 메시지 일련 번호이며, *ll*은 RREQ가 목적지를 향해서 이동할 때 중복되지 않는 레이블이 계속 첨가되면서 형성되는 그룹경로이다. RREQ를 수신하는 노드는 <srcAddr, seqNum>의 조합에 의해서 수신한 RREQ의 중복성을 체크하며, 다음 RREQ 전송 룰에 따라 처리한다.

폐기: RREQ를 수신한 노드가 일반노드이거나 이전에 동일한 RREQ를 수신한 경우에는 재전송 없이 폐기한다.

응답: RREQ를 수신한 목적지 노드는 *ll*을 포함하는 RREP를 생성하고 *ll*의 역경로를 따라 소스에게 전송한다.

전송: 위의 두 룰의 조건을 모두 만족하지 못하면 RREQ를 수신한 노드는 자신의 레이블을 *ll*에 첨부하고 재전송한다.

4.2 패킷 전송

패킷을 가진 노드는 자신의 BL과 그 패킷에 포함되어 있는 그룹경로를 사용하여 다음 노드 즉, 브릿지를 선택한다. 복수의 브릿지가 있으며 패킷을 목적지까지 빨리 이동시킬 수 있는 브릿지를 선택한다.

패킷을 가지고 있는 노드는 먼저 그룹경로상의 자신의 레이블 바로 후에 오는 다음 레이블을 취한다. 자신의 BL에 다음 레이블을 가진 어떤 노드나 다음 레이블을 가진 클러스터를 연결하는 노드가 없는 경우에는 패킷을 자신의 클러스터헤드로 보낸다. 현재 노드가 클러스터헤드이면 자신의 BL를 탐색하여 다음 레이블을 가진 브릿지를 먼저 찾고, 없으면 다음 클러스터레이블을 가진 클러스터에 연결하는 현재 레이블을 가진 브릿지를 선정하여 패킷을 전송한다.

클러스터헤드가 전송할 브릿지를 발견하지 못하는 경우에는 다음 클러스터에 대한 연결이 끊겼다고 판단하고 경로 복구를 실행한다. 경로 복구는 다음 장에서 설명한다. 복구 실패의 경우에는 RERR을 소스 노드에 보내서 그룹경로의 재설정을 요구한다.

링크 파손의 신속한 탐지를 위하여 패킷을 전송하는 노드는 IEEE 802.11 ACK 메커니즘을 이용함으로써 다음 노드로 패킷이 성공적으로 전달되었다는 것을 확인된다. ACK를 수신하지 못한 경우에는 즉각적으로 해당 링크가 파손된 것으로 판단한다.

4.3 그룹경로 복구

그룹경로 상의 두 인접한 클러스터간의 연결이 완전히 파손되는 경우에는 경로를 재탐색하거나 복구하여야 한다. 레이블 *i.j*를 가진 클러스터헤드는 그림 4와 같이 $\text{recoveryTable}_{i,j}$를 사용하여, 자신의 모든 이웃 클러스터들의 각각에 대하여 그것을 거쳐서 갈 수 있는 클러스터들을 관리한다. 예를 들면, 레이블 2.1의 경우에 3.1을 거쳐서 갈수 있는 클러스터는 4.1과 6.1이 된다.

이웃 클러스터의 새로운 생성 혹은 단절을 탐지할 때 마다 변경 내용을 이웃 클러스터헤드들에게 알림으로써 클러스터헤드들은 자신의 테이블을 갱신한다. 그림 4에서 그룹경로상의 2.1과 1.1사이의 연결이 파손된다면, 레이블 6.1에 가기위하여 3.1을 경유하는 우회 경로를 쉽게 얻을 수 있다. 변경된 우회 경로는 RERR을 사용하여 소스 노드에 보고한다.

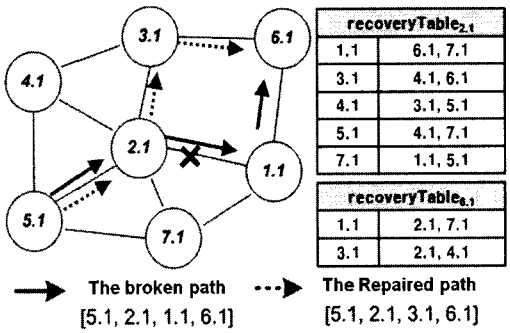


그림 4. 경로 복구를 위한 정보관리

V. DSP 알고리즘의 응용

RREQ를 플러딩 할 때 "RREQ 전송 룰"에 나타난 통제된 재전송을 통해서 오버헤드를 줄일 수 있지만 RREQ가 클러스터에서 클러스터로 이동함에 따라 여전히 불필요한 전송이 많이 발생한다. 예를 들면, 어떤 클러스터의 두 브릿지가 특정 이웃 클러스터를 연결한다고 할 때 하나의 브릿지만 재전송하면 RREQ를 다음 클러스터에 전달할 수 있다. 따라서, 불필요한 전송 문제를 해결하기 위하여 분산 자기 삭제 (DSP: Dsistributed Self-Pruning) 알고리즘 [9]이 제안되었다. DSP 알고리즘에서는 RREQ를 수신한 어떤 브릿지가 자신이 보내고자 하는 모든 이웃 클러스터헤드들에 대하여 자신의 이웃 브릿지가(들이) 동일한 RREQ를 이미 보냈다고 확인이 되면 RREQ를 폐기함으로써 오버헤드를 줄인다. 모든 노드는 자신이 속하는 클러스터와 연결하는 이웃 클러스터들의 각각에 대하여, 클러스터 레이블과 그 클러스터의 클러스터헤드까지의 홉 거리를 저장하는 coverableSet을 관리한다. 클러스터헤드의 경우에 자신과의 거리는 0으로 설정된다. 전송되는 모든 RREQ는 coverableSet을 포함하고, RREQ를 수신하는 노드의 관점에서 이것은 coveredSet이 된다. 즉, 송신자가 RREQ를 보내고 나면 자신이 커버할 수 있는 셋은 바로 커버한 셋이 된다.

그림 5는 수정된 DSP 알고리즘을 설명한다. RREQ를 수신하는 브릿지는 아래 나열한 기준에 따라 정의된 등식 (3)에 의해서 계산된 값을 패킷 재전송 지연 시간으로 타이머를 설정하고 해당 RREQ를 큐에 저장한다.

① 클러스터헤드는 가장 많은 멤버를 커버하기 때문에 전송 지연을 짧게 한다.
② 커버하는 클러스터의 수가 많으면 많을수록 재전송 지연시간을 짧게 한다.
③ 서로 다른 브릿지들에 의한 RREQ의 전송으로 인한 충돌 가능성을 줄이기 위하여 지연시간에 임의의 전송 지연 시간을 설정한다.

기준 ①과 ②로부터 그림 5의 등식 (1)과 (2)가 유도된다. 기준 ③으로부터 random(t)를 등식 (3)에 포함한다. α는 (Radio × fixedTime)이 random(t)에 우세한 형태로 선정되어야 한다.

그림 2의 토폴로지에서 소스 12가 목적지 13까지 경로를 설정하기 위해서 RREQ를 플러딩 하는

```
// RREQ= (srcAddr, seqNum, destAddr, ll, coveredSet)

o Node k that receives RREQ:
  if node k is ORPHAN then
    Free RREQ;
  elseif node k is DESTINATION then
      Create and send RREP;
  else // Decide to relay or drop
      if <srcAddr, seqNum> exists in seenList then
        Free RREQ;
      else
        if node k is CLUSTERHEAD then
          Ratio = 0;                                   (1)
        else
          Ratio = |RREQ.coveredSet| / |coverableSet_k|;   (2)
        endif;
        // random(t) generates a value between 0 and t
        t = α × fixedTime; // 0 < α ≤ 1
        Delay = random(t) + Ratio × fixedTime;         (3)
        Put RREQ into tQueue with timeout = Delay;
      endif;
  endif;

o Node k whose timer expires:
  if <srcAddr, seqNum> does not exist in seenList then
    S = coverableSet_k;
    for each RREQ in tQueue do
      Get element (l, d) from S; // l: label, d: distance
      if (∃ (l, x) ∈ RREQ.coveredSet) such that (x ≤ d)
        S = S - {(l, d)};
      endif;
    endfor;
    if (S != ∅) then
      Relay RREQ with the smallest |RREQ.ll|;
    endif;
    Add <srcAddr, seqNum> to seenList;
    Remove all RREQ from the queue;
  endif;
```

그림 5. 수정된 DSP 알고리즘

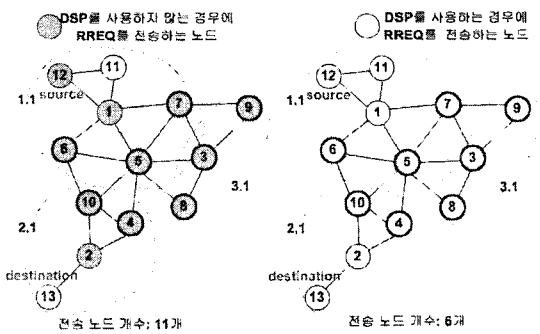


그림 6. DSP의 적용 효과

경우에 전송에 참여하는 노드의 개수를 알아 보기로 하자. 그림 6은 DSP를 적용한 경우와 적용하지 않은 경우에 전송 노드의 갯수를 비교하고 있다. DSP를 사용하지 않는 경우에 모든 브릿지 및 클러

스터헤드가 재전송을 하게 됨으로 11개의 노드가 전송에 참여한다. DSP를 사용하는 경우에는 6개의 노드만이 전송에 참여하게 된다.

VI. 성능 평가

그룹 경로의 안정성과 패킷처리 효율성을 평가하기 위하여 패킷 전송률, RERR 발생 개수, 라우팅 오버헤드, 패킷 전송지연, 지터 등의 성능 평가지표를 사용하였다. 경로 안정성을 평가하기 위하여 노드의 이동속도를 변경시키고, 처리 효율성을 평가하여 위하여 트래픽을 변경하였다.

NS2 시뮬레이터에서 2 Mbps IEEE 802.11 MAC과 Random Waypoint Model을 사용하였다. 각 시나리오에 대하여 10번의 시뮬레이션을 수행하여 얻은 값들의 평균값을 제시한다. 오버헤드는 HELLO, RREQ, RREP, RERR 등 데이터 패킷의 전송을 제외한 모든 메시지를 포함하였다. 공통으로 사용된 파라메타 값은 테이블 1과 같다. DSP 알고리즘을 적용한 GDSR는 GDSR_DSP로 표기하였다.

표 1 시뮬레이션 파라메타

파라메타	값
시뮬레이션 시간	500s
영역 크기	1500 x 1500 (m^2)
노드 수	100
전송거리	250m
패킷 크기	400bytes
쉼 (Pause) 시간	30s

6.1 노드 이동성 변화에 따른 GDSR_DSP

최대 노드 이동 속도를 0, 5, 10, 15, 20, 25m/s로 변화시키면서, 2 packets/s 속도로 패킷을 전송하는 50개의 CBR 세션을 사용하였다.

그림 7은 소스에서 발생된 패킷들 중 목적지 노드까지 성공적으로 도착한 전송 성공률에 대한 비교이다. 먼저, 2-홉 토폴로지 정보를 사용하여 패킷을 구제하는 CBRP의 전송률이 DSR의 그것보다 다소 높다는 것을 알 수 있다. 또한, GDSR_DSP는 다른 두 프로토콜에 비해서 속도 변화에 대하여 훨씬 덜 민감하다는 것을 알 수 있으며, 최대 속도 변화에 대하여 우수한 성능을 나타낸다는 것을 알 수 있다. 속도에 대한 전송률의 민감도가 차이가 나는 원인은 경로의 안정성 때문이다. 노드 이동성이 높아지면 링크의 파손 확률이 높아지고, 그 결과 단일

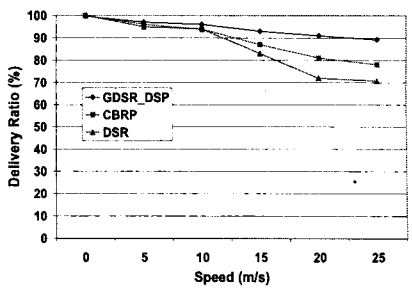


그림 7. 전송율 비교

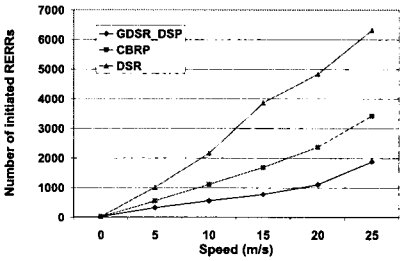


그림 8. RERR 생성 개수

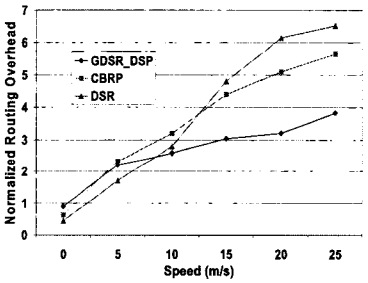


그림 9. 라우팅 오버헤드

링크 파손에 크게 좌우되는 노드경로를 사용하는 DSR과 CBRP의 경로 안정성이 크게 떨어짐을 알 수 있다. 따라서, DSR 및 CBRP의 전송률이 빠르게 감소함을 알 수 있다. 반면에, 그룹경로는 단일 링크 파손에 덜 민감하기 때문에 안정된 성능을 보여준다. 라우팅 경로가 깨어졌을 때 발생하는 RERR 메시지의 개수가 경로 안정성을 대변 할 수 있는 지표이며, 그림 8에서 GDSR_DSP가 RERR을 훨씬 적게 생성한다.

이것은 목적지에 성공적으로 전달된 각 패킷에 대한 소요된 콘트롤 메시지의 수의 비율로써 정의되는 정규화된 라우팅 오버헤드에도 크게 영향을 미친다. 그림 9을 보면 DSR의 라우팅 오버헤드가 이동성 속도가 증가함에 따라 가파르게 상승함을 알 수 있다. DSR의 경우에 경로 재 탐색을 요구하는 RERR을 가장 많이 생성하고 (그림 8 참조), 경

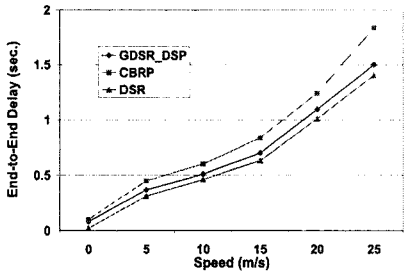


그림 10. 양극단 지연시간

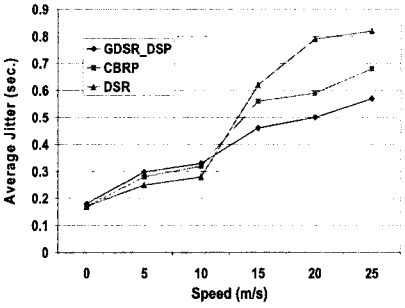


그림 11. 전송 지터 (Jitter)

로 탐색에서도 통제되지 않는 RREQ 플러딩을 사용하기 때문이다. 이러한 결과를 볼 때 노드 이동성 변화에 대한 GDSR_DSP의 적응력이 뛰어나다는 것을 알 수 있다.

그림 10을 보면, 세 개의 프로토콜의 양극단 지연시간은 모두 노드 최대 이동 속도에 비례하여 증가하는 것을 알수 있다. 지연시간은 목적지에 성공적으로 도착한 패킷들 만을 대상으로 계산하였기 때문에 상대적으로 전송시간이 길어질 수 있는 전송실패한 패킷들이 많은 DSR의 경우 다소 전송 지연시간이 낮음을 알 수 있다. 지터값은 패킷의 도착 시간들 간의 간격들의 분산으로 통신 흐름의 일관성을 나타내어 특히 멀티미디어 통신 시 중요한 성능척도가 된다. 그림 11을 보면 GDSR_DSP가 낮음 지터를 가지고 있음을 알 수 있다. 경로 재탐색은 패킷 전송 지연의 변화에 큰 영향을 미치기 때문에 경로 재탐색의 빈도가 낮은 GDSR_DSP가 낮은 지터를 보여준다는 것은 자명한 결과이다.

6.2 트래픽 부하의 변화에 다른 비교

최고 이동속도를 10m/s로 고정한 상태에서 2 packets/s를 갖는 CBR 세션의 수를 변화시키면서 성능을 측정하였다. 그림 12의 전송률 비교를 보면 안정적인 그룹경로를 사용하는 GDSR_DSP가 높은 가용

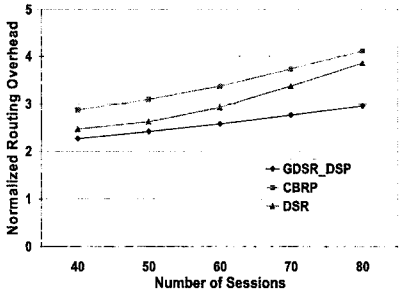


그림 12. 전송율

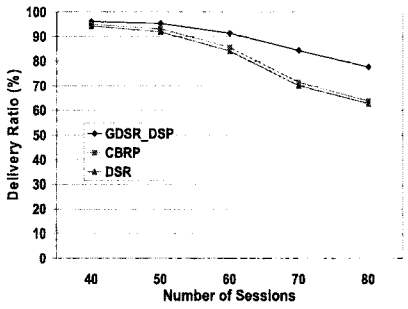


그림 13. 라우팅 오버헤드

대역폭으로 인하여 과도한 네트워크 트래픽에도 적응력이 뛰어나다는 것을 알 수 있다. 그림 13을 보면 DSR은 주기적인 전송 패컨을 갖는 Hello 메시지를 사용하지 않음에도 불구하고 오버헤드가 가장 높음을 알 수 있다. 이러한 이유는 그림 8에서 나타난 결과와 같이 경로 재탐색의 빈도가 높기 때문이다.

6.3 DSP의 효과

마지막으로 GDSR에 DSP의 적용 효과를 분석하기 위하여 시뮬레이션을 수행하였다. 1000 x 1000 (m^2)의 고정된 영역에서 노드 밀도를 변화시키면서 DSP의 효과를 검토한다. 각각 5 packets/s를 갖는 30개의 CBR 세션을 사용하였으며, 최대 이동 속도를 10m/s로 고정하였다.

그림 14을 보면, DSP 알고리즘을 사용하는 경우에 노드들의 RREQ 재전송이 효과적으로 억제됨을 알 수 있으며, 노드의 수가 100개일 때 RREQ 재전송에 참여하는 노드의 수가 60% 이상 감소한다는 것을 알 수 있다. 그림 15를 보면, DSP를 채택하지 않은 GDSR은 노드 밀도가 높아 짐에 따라 RREQ 재전송에 참여하는 노드의 수가 크게 증가하고, 그결과 오버헤드가 급격히 증가함을 알 수 있다. 반면에 DSP 알고리즘을 사용하는 경우에 오버헤드가 완만하게 증가함을 알 수 있다.

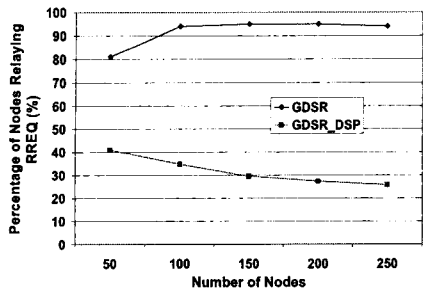


그림 14. RREQ를 재전송하는 노드의 수

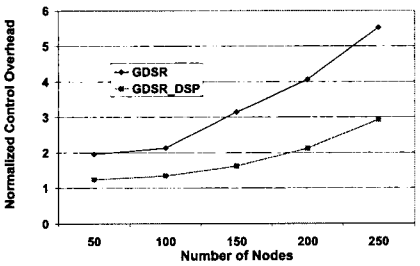


그림 15. 전송율

Ⅶ. 결 론

복수경로를 의미하는 그룹경로는 다른 두 라우팅 프로토콜에서 사용하는 노드경로 보다 단일 링크 파손에 덜 민감하고 안정적이다. 그룹 경로의 안정성은 GDSR이 경로 재탐색의 수와 밀접한 관계가 있는 RERR을 훨씬 덜 생성한다는 것을 통해서 입증되었다. 또한, 메시지 플러딩 과정에서 불필요한 전송을 줄이기 위해서 고안된 DSP 알고리즘은 GDSR에 효과적으로 적용하였으며, 그 결과로 콘트롤 오버헤드를 약 40% 정도 개선하였으며, 노드 밀도가 높아짐에 따라 개선의 폭이 더욱 증가함을 알 수 있었다. 결론적으로, GDSR은 노드 이동속도, 트래픽 등의 변화에 대하여 적응력이 뛰어남을 확인할 수 있었다.

참 고 문 헌

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

[2] Z. Haas, M. Pearlman, and P. Samar, "Zone routing protocol," IETF Internet Draft, draft-ietf-manet-zrp-04.txt, July 2002.

[3] D. Johnson, D. Maltz, Y Hu and J Jetcheva, "The Dynamic Source Routing protocol for Mobile Ad Hoc Networks", IETF MANET Internet Draft, Feb 2002.

[4] M. Jiang, J. Li, and Y. C. Tay, "Cluster bsed routing protocol (CBRP) functional specification," IETF Internet Draft, draft-ietf-manet-cbrp-spec-01.txt, July 1999.

[5] V. Li, H. S. Park and H. Oh, "A cluster-label based mechanism for backbones on mobile ad hoc networks," Springer Verlag, LNCS 3970, pp.26-36, May, 2006.

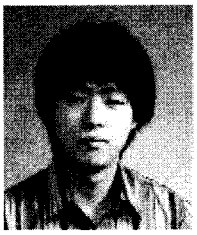
[6] H. Oh and S. Y. Yun, "Proactive cluster-based Distance Vector Routing Protocol for Mobile Ad Hoc Networks," IEICE Transactions on Communications, Vol. E89-B, Jun 2007.

[7] C. Perkins and P. Bhagwat, "Highly dynamic destination sequenced distance vector routing for mobile computers," In Proceedings of ACM SIGCOMM, 24(4), Oct 1994.

[8] C.E. Perkins and E. M. Royer, "Ad hoc On-demand Distance Vector Routing," http://tools.ietf.org/html/draft-ietf-manet-aodv-09, IETF, Internet Draft.

[9] S. Y. Yun and H. Oh, "Distributed Self-Pruning (DSP) Algorithm for Bridges in Clustered Ad Hoc Networks," LNCS 4523, pp.132-140, 2007.

곽 운 용 (Woon Yong Kwak) 준회원



2008년 울산대학교 정보통신공학전공 학사

2008년 울산대학교 학석사 통합과정

<관심분야> 애드 혹 네트워크 프로토콜, 센서 네트워크, 임베디드시스템

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 부교수

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