논문 12-37B-10-01

한국통신학회논문지 '12-10 Vol.37B No.10
http://dx.doi.org/10.7840/kics.2012.37B.10.859

인프라구조 기반의 이동 애드혹 네트워크를 위한 인터넷 게이트웨이 중심의 링크상태 라우팅 프로토콜

이 성 욱◆, 오 지 충*, 한 충 진*, 김 제 욱*, 오 훈°

An Internet Gateway Based Link State Routing for Infrastructure-Based Mobile Ad Hoc Networks

Sung Uk Lee◆, Chi-Trung Ngo*, Trung-Dinh Han*, Je-Wook Kim*, Hoon Oh°

요 약

인프라구조 기반의 이동 애드혹 네트워크를 위한 기존의 프로토콜들은 이동성 관리와 라우팅 프로토콜을 분리하여 설계되었을 뿐만 아니라 플러딩을 사용하기 때문에 높은 제어 오버헤드를 발생시킨다. 본 논문에서는 트리 기반의 이동성관리 방식을 사용하며 이 과정에서 부수적으로 구축되는 토폴로지 정보를 활용하는 단순하고 효율적인 라우팅 프로토콜을 제시한다. 제안하는 라우팅 프로토콜은 패킷 전송 과정에서 트리를 구성하는 중요 링크에 대한 파손을 발견하는 경우에 토폴로지 정보를 신속히 갱신한다. 이러한 방식으로 이동성 관리와 라우팅 프로토콜은 서로 협력한다. 또한 IG 주위에 트래픽 혼잡을 줄이기 위하여 점진적 경로탐색 방식을 사용하였으며, 제어메시지 전송 신뢰성을 높이고 링크 유효성을 신속히 판단할 수 있도록 유니케스트 기반의 브로드케스트 방식을 사용하였다. 그리고 플러딩을 배제하고 제어메시지 전송의 최적화를 통하여 오버헤드를 크게 줄임으로써 노드의 수가 증가하고 이동성이 높은 경우에도 안정적으로 작동할 수 있도록 하였다. 시뮬레이션을 통해서 제안하는 프로토콜이 Hybrid 이동성 관리 방식을 사용하는 AODV에 비하여 우수한 성능과 확장성을 갖는다는 것을 입증하였다.

Key Words : tree topology, ad hoc networks, mobility management, link state protocol, shortest path

ABSTRACT

Since the existing protocols separated mobility management part and routing protocol part in their design and used a flooding, they suffer from the high control overhead, thereby limiting performance. In this paper, we use a tree-based mobility management method and present a simple and efficient routing protocol that exploits the topology information which is built additionally through mobility management. Thus, the mobility management and the routing protocol closely cooperate to optimize control overhead. Furthermore, we use a progressive path discovery method to alleviate traffic congestion around IG and a unicast-based broadcast method to increase the reliability of message delivery and to judge link validity promptly. The proposed protocol reduces control overhead greatly and works in a stable manner even with the large number of nodes and high mobility. This was proven by comparing with the AODV protocol that employs the hybrid mobility management protocol.

※ 본 연구는 울산시 및 교육과학기술부의 지역과학연구단지 개발 프로그램의 지원으로 수행되었습니다.
◆ 주저자 : 포항산업과학연구원, 정회원
° 교신저자 : 울산대학교 전기공학부 조선해양IT융합연구실, hoonoh@ulsan.ac.kr, 종신회원
* 울산대학교 전기공학부 조선양IT융합연구실
논문번호 : KICS2012-09-422, 접수일자 : 2012년 8월 23일, 최종논문접수일자 : 2012년 10월 12일

Ⅰ. 서 론

이동 애드혹 네트워크 (MANET)은 전개 및 관리의 용이성으로 인하여 응용 범위가 넓다. 하지만, MANET에 속한 노드들이 인터넷 엑세스 혹은 다른 MANET에 속한 이동노드와 통신을 원하는 경우에 큰 제약을 받는다. 기지국 (BS: Base Station, 본 논문에서는 IG: Internet Gateway라고 칭함) 및 이동 노드 (MN: Mobile Node)로 구성되는 인프라구조 기반의 MANET (IFMANET: Infrastructure-based MANET)에서는 IG가 자기로부터 서비스를 필요로 하는 MN들의 이동성을 관리해야 한다. 이러한 이동성 관리는 무선 1홉 만을 허용하는 IFMANET [2] 에서는 어렵지 않지만, 무선 멀티홉 MN들을 갖는 IFMANET에서는 높은 오버헤드를 유발한다. 더구나, 라우팅 프로토콜로는 MANET를 위해서 설계되었던 DSDV [4], AODV [3], DSR [5], OLSR[15]를 그대로 사용하고 있다. 하지만, IFMANET에서는 고성능, 고속 유선 대역폭, 그리고 이동성 관리에서 IG가 수집한 토폴로지 정보를 활용할 수 있기 때문에 새로운 라우팅 프로토콜의 설계가 필요하다.

대부분의 기존의 방식은 이동성 관리의 효율성을 개선하는데 중점을 두었다. Tseng et al. [18]은 IEEE 802.11 기반의 무선 네트워크와 MANET를 통합하기 위한 IFMANET 아키텍쳐를 제공했다. Broach et al. [11]은 IG가 두 개의 네트워크 인터페이스 카드를 가지도록 하였다. 하나는 표준 IP로 인터넷 브릿지 역할을 하고, 다른 하나는 MN 브릿지로 역할을 수행한다. 라우팅을 위하여 DSP 프로토콜을 사용하였다. Jonsson el al. [6]은 Cell Switching 알고리즘을 사용하여 MN이 IG에 등록하는 타이밍을 최적화하는데 중점을 두었으며, AODV 라우팅 알고리즘을 사용하였다. Sun et al. [19]은 Mobile IP와 AODV가 MN과 IG 사이에서 멀티홉 경로를 찾는데 있어서 협력 방법을 논의하였다. Ratanchandani et al. [17]은 에이전트 광고, TTL 범위 설정, 에이젼트 광고 듣기, 엿듣기, 에이젼트 응답요청 등과 같은 기법들을 통합하는 하이브리드 기법을 제안하였다. Ammari et al. [7]은 DSDV [4]를 사용하였으며, MN과 IG 사이에 이동 게이트웨이를 사용하는 3-계층 방법을 제안하였다. 더구나, Moshrify et al. [8]과 Ruiz et al. [16]은 AODV에 기반한 게이트웨이 탐색 기법을 제안하였다.

위에 열거한 방식들은 이동성 관리기법을 개선하는데 치중 하였지만, 플러딩을 사용하기 때문에 그 효과가 제한적이다. 더구나 이동성 관리와 라우팅프로토콜이 둘 다 플러딩을 사용하는 경우에는 성능이 더욱 악화된다. 이러한 방식들과는 달리 트리기반의 접근 방식 [10]에서는 트리 관리 오버헤드를 줄이기 위하여 다수의 소규모의 트리를 관리하는 방법을 제시하였다. 하지만, 경로 탐색을 위하여 플러딩의 범위를 제한하기는 하지만 여전히 플러딩을 사용한다.

본 접근 방식은 조상 노드가 관리하고 있는 토폴로지 정보에 기반하여 경로를 탐색한다는 점에 있어서 다른 트리 기반의 접근방식 [10, 14, 21]과는 다르다, 하지만, 노드들이, 부분이든 전체이든, 네트워크 토폴로지 정보를 관리한다는 점에 있어서 LS[13], FISHEYE[1], PCDV[9], OLSR[15]과 같은 링크상태 라우팅 프로토콜과 유사하다. IFMANET에 속하는 모든 노드들은 IG에서 시작하는 트리를 구성한다. 각 노드는 자신의 1-홉 링크상태를 트리경로를 따라서 주기적으로 IG에 전송하도록 한다. 각 노드는 자신 및 후손 노드들에 대하여 부분적인 네트워크 토폴로지를 관리하는 반면에 IG는 전체 네트워크 토폴로지를 관리할 수 있다. 경로를 얻고자 하는 노드는 IG에 경로계산요청 메시지를 보낸다. IG는 토폴로지 정보를 사용하여 경로를 계산하고, 계산된 경로를 그 노드에 전달한다. 이러한 경로 탐색은 IG가 정확한 토폴로지 정보를 유지한다면 최단경로를 획득할 수 있다는 장점이 있다. 하지만, 경로 획득의 시간이 증가하고, IG에 더 가까운 노드들은 트래픽 혼잡현상을 겪는다. 이 문제를 개선하기 위하여 점진적인 경로 탐색방법을 제안하였다. 즉, 소스 노드를 포함하여 IG까지의 경로상에 있는 모든 노드는 경로계산요청 메시지를 받을 때 마다 경로를 계산하고 경로를 찾는 즉시 소스에 알려준다.

이 방식은 몇 가지 특징이 있다. 트리 기반의 이동성 관리를 통해서 구축된 토폴로지 정보를 이용하여 경로 탐색을 쉽게 할 수 있으며, 이동성 관리와 경로 탐색에서 플로딩을 사용하지 않는다. 모든 노드가 이웃 노드 정보를 알고 있기 때문에 유니케스트 기반의 브로드케스트 방식을 사용할 수 있다. 따라서, 전송신뢰성이 높기 때문에 파손된 링크를 빠르고 정확하게 감지할 수 있다. 또한, TG (Topology Graph)의 정확성이 증가되고 경로의 유효성이 증가하기 때문에 결과적으로 네트워크 오버헤드가 줄어든다. 트리 관리에서 발생하는 루프 문제를 해결하기 위해서는 루프 감지 및 해결 (LDR) 방식 [20]를 사용하였다. 시뮬레이션을 통해서 제안된 프로토콜은 높은 노드이동성을 가진 네트워크 환경에서도 신뢰성과 확장성이 뛰어나다는 것을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 네트워크 모델, 필요한 정의, 그리고 프로토콜 제안 동기를 설명하는 본 논문의 배경을 기술한다. 3장에서는 제안하는 프로토콜을 상세히 기술한다. 4장에서는 다양한 시뮬레이션 시나리오를 제시하고 합리성 평가 및 성능평가지표를 제시하고 다양한 시뮬레이션 결과에 대하여 논의한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 연구배경

2.1. 네트워크 모델

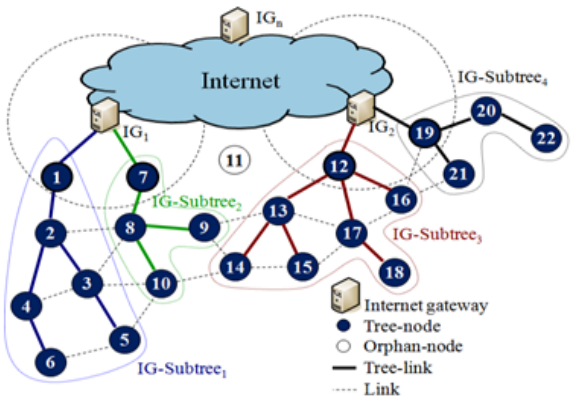

그림 1. 네트워크 모델
Fig. 1. Network model

네트워크는 복수의 트리를 형성하는 IG와 MN들로 구성되고, 각 트리는 IG에서 시작한다. 모든 노드는 다른 노드가 전송하는 패킷을 엿들을 수 있으며 [12], IG들은 높은 대역폭을 갖는 유선 통신망을 사용하여 토폴로지 정보를 쉽게 공유할 수 있다고 가정한다. 또한, 모든 노드는 통신을 시도할 의향이 없다고 할지라도 IG에 등록을 유지해야 한다. 그 이유는 원격 호스트 혹은 다른 MANET에 속한 MN이 인터넷을 통하여 연결을 요청할 수 있기 때문이다.

네트워크에 속한 노드들 (IG 혹은 MN)과 노드들 사이의 연결의 집합은 비방향성 그래프를 형성한다. 토폴로지 그래프는 토폴로지 그래프를 형성하는 과정에서 IG를 기준 노드로 하는 Minimum Spanning Tree를 형성할 수 있다. 트리에 속하는 노드는 트리노드, 자식노드를 가지고 있다고 할지라도 부모노드를 가지고 있지 않는 노드는 고아노드라고 한다. 특히, 부모와 자식노드 사이의 링크를 트리링크라고 하고, IG의 자식노드를 루트로 하는 서브트리를 IG-Subtree라고 칭한다. 그림 1을 보면, 두 개의 트리가 IG1과 IG2로부터 형성되어 있으며, 실선은 트리링크, 트리에 속하지 않는 점선은 일반적인 링크를 나타낸다. 네 개의 IG-Subtree는 노드 1, 7, 12, 19에서 형성된다.

2.2. 표기 및 메시지 정의

본 논문에서 사용하는 주요 표기는 다음과 같다.

- *ID* : 노드의 고유번호
- *HopToIG* : 노드로부터 IG까지 홉 거리
- *i.TS* : 루트가 노드 i인 어떤 서브트리에 속하는 트리 노드의 집합
- *i.NS* : 노드 i의 이웃 노드의 집합

다음과 같이 제어 메시지를 정의하여 사용한다.

- *Hello = (ID, HopToIG)* : IG는 주기적으로 자신의 트리정보에서 자신의 자식노드 중의 하나를 선정하여 HopToIG = 0인 Hello 메시지를 전송한다. 자신의 트리정보에 자식노드가 없는 경우에는 Hello 메시지를 브로드캐스트 한다. MN은 Hello 메시지를 자신의 부모에게 주기적으로 전송한다. 다른 노드는 엿듣기를 사용하여 Hello 메시지를 수신한다.
- *J-REQ = (ID)* : MN은 자신으 부모로 삼고 싶은 노드에게 J-REQ(Join Request) 메시지를 보낸다.
- *CR-REQ = (ID)* : 고아노드는 자신의 자식노드에게 "논리적 떠나보내기"[1]를 요청하기 위하여 CR-REQ (Children Leave Request) 메시지를 자식노드에게 보낸다.
- *LU-REQ = (ID, NS)* : MN은 규칙적으로 IG에게 LU-REQ (Link Update Request) 메시지를 보낸다. 이 메시지를 수신하는 중간 노드는 자신의 토폴로지 정보를 갱신한다.

2.3. 토폴로지 그래프 (TG)

2.3.1. TG 정의

모든 노드 i는 다음과 같이 자신의 토폴로지 그래프 i.TG를 갖는다:

$$i.TG = (i.V, i.E),$$

여기서 $i.V = \{x, y | x \in i.TS, y \in x.NS\}$, $i.E = \{(x, y) | x \in i.TS, y \in x.NS\}$.

본 정의에서 i.V는 i.TS에 속하는 모든 노드와 각 노드들의 이웃 노드를 포함한다. i.E는 i.TS에 속하는 모든 노드들이 만드는 링크들과 i.TS에 속하는 각 노드와 이웃 노드들과의 링크들을 포함한다. 노드 i가 IG이면, i.TG는 전체 토폴로지 그래프가 된다. 트리

1) "논리적 떠나보내기"는 어떤 노드가 더 이상 부모자식 관계를 유지하지 않고 링크 만을 유지하는 상태를 말한다.

노드는 IG에 경로 요청 메시지를 전송함으로써 경로를 얻을 수 있다. 예를 들면, 그림 1에서 4.TS = {4, 6}, 4.TG = ({4, 6, 2, 3, 5}, {(4, 2), (4, 3), (4, 6), (6, 5)}) 이다.

2.3.2. TG 형성 및 관리방식

트리노드는 자신의 부모에게 주기적으로 Hello 메시지를 보낸다. 반면에 IG는 자신의 자식 노드 중의 하나에게 Hello 메시지를 주기적으로 보내고, 모든 다른 이웃 MN은 Hello 메시지를 엿듣기 함으로써 자신의 NS를 관리한다. 모든 트리노드는 주기적으로 자신의 링크상태정보 (NS)를 포함하는 LU-REQ 메시지를 트리 경로를 따라서 IG에 전송한다. 이러한 제어메시지를 수신함으로써 노드 i는 i.TS를 관리하고 i.TS에 속하는 각 노드의 링크상태를 관리한다. 따라서, MN은 부분 토폴로지 그래프를 관리할 수 있으며, IG는 네트워크의 전체 토폴로지 그래프를 관리할 수 있다.

트리노드는 기본적으로 IG로부터 경로를 획득한다. 하지만, 링크상태 경신 지연으로 인하여 TG가 항상 최신의 정보를 관리할 수 없기 때문에 획득한 경로가 이미 유효하지 않을 수 있다. TG의 정확성은 각 MN이 얼마나 빠르고 정확하게 링크 상태를 탐지하고 얼마나 자주 IG에 자신의 링크상태를 보고하는가에 달려 있다. U-REQ메시지의 전송주기를 짧게 하면 제어 오버헤드가 증가한다. 본 논문에서는 오버헤드를 줄이고 라우팅의 정확성을 개선하기 위하여 링크의 특성(트리링크, 일반링크)에 따라서 링크관리를 차별화 한다.

트리 링크는 제어 메시지를 전달하기 때문에 일반 링크보다 더 중요하다. 더구나, 어떤 트리노드가 유효한 트리링크를 파손이라고 오판하면, 불필요하게 재가입 프로세스를 거치게 될 것이다. 판단의 정확성과 속도를 개선하기 위하여 유니케스트 기반의 브로드케스트 (uBroadcast) 방식을 이용한다. uBroadcast는 RTS, CTS, ACK와 같은 MAC 계층 제어메시지를 사용하기 때문에 제어오버헤드를 다소 증가 시킨다고 할지라도 uBroadcast 방식이 전송 신뢰성을 크게 증가시킨다는 것을 시뮬레이션을 통해서 확인하였다 (4장 평가에서 제시). 시뮬레이션 결과에 의하면, 상당히 고밀도 및 높은 트래픽 네트워크에서도 목적지 노드에 대한 전송 성공률이 거의 100%, 엿듣는 노드에 대한 전송성공율이 80% 이상을 달성하였으며, 브로드케스트의 평균 전송성공율 70%정도에 비해서 전송 성공률이 크게 향상 되었다. 유니케스트 방식을 사용함으로써, 어떤 노드가 한 주기 내에 LU-REQ를 수신하지 못하면 자신의 TS에서 해당노드를 삭제한다. 이는 보통 2 주기의 실패마다 삭제를 하는 기존의 방식에 비해서 크게 개선되었다고 할 수 있다. Hello 메시지도 uBroadcast를 사용하기 때문에 동일하게 NS 관리에 사용된다.

2.4. TLSR 프로토콜의 제안 동기

IFMANET에서 프로토콜의 효율성 (PE : Protocol Efficiency)은 크게 이동성관리 (MM: Mobility Managemnt)의 효율성 (MME : MM Efficiency)과 라우팅 (RT: Routing)의 효율성 (RTE : RT Efficiency)에 의해서 결정된다. 즉, 어떤 프로토콜 A의 효율성은 다음과 같이 나타낼 수 있다.

$$PE(A) = MME(A) + RTE(A)$$

IFMANET를 위한 기존의 프로토콜들은 MME와 RTE를 분리해서 논의하였으며, MME에 치중을 하고 RTE는 프로토콜에 따라서 다소 차이가 있기는 하지만 기존의 프로토콜을 사용하였다. 서론에서 논의 하였듯이 여러 가지 이동성 관리 프로토콜이 제안되었으나 크게 보면 세 가지 카테고리, 즉 Proactive, Reactive, Hybrid 방식[17] 중의 하나에 포함된다고 볼 수 있다. 이러한 방식의 장점은 MME와 RTE를 분리해서 생각할 수 있으며, RTE를 위해서 기존에 MANET를 위해서 제안된 성능이 검증된 프로토콜을 약간의 수정 후에 그대로 사용할 수 있다는 장점이 있다. 하지만, 이동성 관리를 위하여 플러딩을 사용하고 있으며, AODV, DSR과 같은 요구기반 라우팅 프로토콜을 사용하는 경우에는 라우팅에서도 플러딩을 사용하게 됨으로써 제어 오버헤드 문제가 악화된다.

최근에 트리기반의 이동성 관리 방식[10] 및 IG 주위에 제어 메시지의 트래픽 증가로 인한 병목현상을 해결하는 방식[22]을 제공함으로써 MME를 크게 개선할 수 있다는 것을 입증한 바 있다. 전체 트리 구성은 일회적이고 유지보수는 필요시만 지역적으로 수행하기 때문에 전체 비용은 주기적인 플러딩에 비해서 아주 낮다. 또한, 복수의 작은 트리를 구성하거나 트리들의 크기를 균형적으로 관리함으로써 오버헤드를 더욱 개선할 수 있다. 이 방식에서는 각 노드는 자신의 등록 메시지 (Registration Message)를 주기적으로 혹은 필요한 경우에 트리 경로를 따라서 IG에 전달한다.

중요한 것은, 등록메시지에 추가적인 노드의 링크 상태정보를 실어서 보냄으로써 IG는 전체 토폴로지

정보를 관리하고, 중간 노드들은 자신과 자신의 후손 노드들에 대한 부분 토폴로지 정보를 관리할 수 있다. 본 논문에서는 이러한 추가적인 링크상태정보 (NS)를 가진 등록 메시지를 LU-REQ 라고 한다. 이렇게 이동성 관리를 하는 과정에서 부수적으로 구축된 토폴로지 정보는 라우팅 프로토콜을 설계하는데 매우 효율적으로 사용될 수 있다. 모든 노드는 자신(소스)과 통신을 하고자하는 노드 (목적지)를 IG에 보내면 쉽게 경로를 획득할 수 있다.

라우팅 프로토콜이 단순하다는 점에 있어서 장점이 있다. 하지만, 라우팅 성능은 어떻게 토폴로지 정보를 최신으로 유지할 수 있는가에 크게 의존한다. 각 노드가 보내는 LU-REQ의 전송 간격을 작게 하면 토폴로지 그래프의 정확도는 개선되지만 오버헤드가 증가하고, 전송간격을 크게 하면 토폴로지 그래프의 정확도가 떨어지고, 어떤 노드가 획득한 경로가 자주 유효하지 않을 수 있다.

표 1. TLSR과 기존프로토콜의 비교
Table 1. TLSR and existing protocols

Existing protocols	
MM	- use proactive, reactive, and hybrid variations - Gateway discovery using flooding
RT	- Modified version of MANET protocols - Almost independent of mobility management protocol - Use flooding (Reactive) or share topology information (Proactive)
TLSR	
MM	- Registration along tree paths - Topology information construction and update - No flooding - Quicker and more accurate management of link information using uBroadcast
RT	- Acquire a path reactively using topology information that is built additionally in mobility management - Update topology information through topology management and maintenance process - No flooding

이를 개선할 수 있는 방법으로 각 노드가 경로 획득 혹은 경로 재설정을 위하여 메시지를 보낼 때 자신의 링크상태정보를 피기백하여 보내거나, 라우팅 및 데이터 전송과정에서 발견한 중요도가 높은 트리링크에 대해서는 긴급갱신을 함으로써 개선할 수 있다. 이것은 이동성 관리와 라우팅 프로토콜이 서로 협력하는 모델로 기본의 방식들과는 기본적으로 차이가 있다고 할 수 있다. 또한 각종 제어메시지를 전송에서 uBroadcast 방식을 사용함으로써 링크정보 변경에 대한 유지보수의 정확도를 높일 수 있다. 표 1에서 기존 프로토콜과 TLSR의 차이점을 요약하였다.

Ⅲ. 트리링크 상태 라우팅 (TLSR: Tree Link State Routing) 프로토콜

3.1. 토폴로지 관리

3.1.1. MN-IG 결합 프로세스

고아노드는 IG가 자식노드에 보내는 Hello 메시지를 엿듣는 경우에 IG에 J-REQ를 보낸다. ACK를 수신하면 IG의 자식노드가 되고 HopToIG = 1로 설정한다. J-REQ를 수신하는 IG는 해당 노드를 자식노드로 취한다.

3.1.2. MN-MN 결합 프로세스

고아노드가 어떤 트리노드가 자신의 부모에게 보내는 Hello 메시지를 엿들으면 다른 트리노드로부터 더 많은 Hello 메시지를 수신하기 위하여 잠시 동안 (대기시간 < Hello_interval) 대기한다. 대기 시간이 만기되면 IG까지 가장 짧은 거리를 제시하는 트리노드를 선택하고 결합하기 위하여 J-REQ를 보낸다. 해당 고아노드가 ACK를 수신하면 IG까지의 거리를 설정하고 결합은 완료된다.

3.1.3. 부모 변경 프로세스

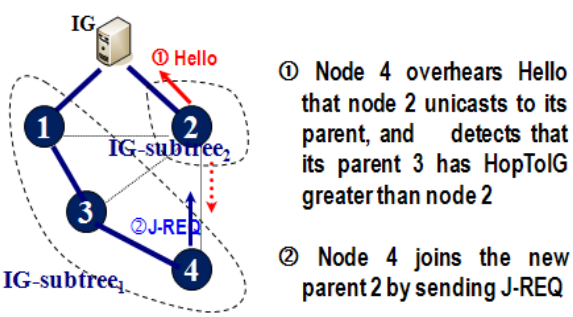

그림 2. 부모 변경 프로세스
Fig. 2. Parent change process

어떤 트리노드가 Hello 메시지를 엿들을 때 자신의 부모노드가 Hello 메시지를 송신한 이웃 노드보다도 더 큰 HopToIG를 가지고 있다면 해당 트리노드는

J-REQ를 보내서 즉각 이웃 노드로 부모를 변경한다. 여기서, 자신의 이전 부모노드에게 자신이 떠나는 것을 알리지 않는다. 이전 부모노드는 Hello 메시지 간격 혹은 LU-REQ 간격 내에 Hello 메시지나 LU-REQ를 수신하지 않으면 자신의 TS로부터 제거한다. 앞에서 설명하였듯이 uBroadcast를 사용하기 때문에 신속하게 링크상태변화를 판단하고 갱신한다. 그림 2는 노드 4가 새로운 부모에게 결합하는 과정을 보여준다.

3.1.4. 트리링크 파손복구 프로세스

어떤 노드가 한 Hello 간격 (Hello_Interval) 동안 어떤 자식노드로부터 Hello 메시지를 받지 않으면 해당 링크가 파손되었다고 판단한다. 또한, 어떤 노드가 부모노드에게 Hello 메시지를 보낸 즉시 ACK를 받지 않으면 해당 링크는 파손되었다고 판단한다. 부모노드에 대한 링크 파손을 감지한 노드는 고아노드가 되고 이웃 목록에서 가장 짧은 HopToIG를 가진 노드를 선택하여 결합을 시도한다. 이 때, 루프를 방지하기 위해서 자신의 후손 노드를 선택하지 않는다. 이러한 과정은 결합을 성공할 때까지 혹은 더 선택 가능한 이웃 노드가 없을 때까지 계속된다.

고아노드가 새로운 부모노드를 발견하지 못하는 경우에는 자신의 후손 노드들을 유지 한 체 다른 행위를 하지 않는다. 그 이유는 동적 네트워크에서 새로운 부모를 곧 발견할 수도 있기 때문이다. 만일 해당 고아노드가 자식 노드에게 떠나서 다른 부모노드를 각자 찾으려 한다면 많은 오버헤드를 발생시키게 된다. 이러한 일이 후손 노드들에게 연속적으로 일어나면 제어오버헤드는 크게 증가할 것이다. 하지만, 자식노드가 메시지를 보내서 메시지 전송 서비스를 요청한 경우에만 부모를 잃은 상황을 알려주고 스스로 부모 변경을 할 수 있도록 유도한다.

```
At src x that wants to find dst z:
  set PC-RES-timer; //to wait for a response
  send PC-REQ to its IG;
  wait for PC-RES until  PC-RES-timer expires;
  IF PC-RES-timer expired THEN return; ENDIF
  IF PC-RES arrives THEN
     initiate packet transmission with the path;
  ENDIF

At an IG y that receives PC-REQ:
  calculate a path from y.TG by using the SP algorithm;
  IF path is found THEN
    reply PC-RES with path = (src, …, dst) to src;
  ENDIF
```

그림 3. SP(최단 경로) 탐색 알고리즘
Fig. 3. SP discovery algorithm

3.2. 경로관리

3.2.1. SP (최단 경로) 탐색 방식

IG는 전체 토폴로지 정보를 가지고 있기 때문에 주어진 소스노드 및 목적지 노드에 대하여 최단 거리를 구할 수 있다. 따라서, 소스노드는 IG에게 PC-REQ 메시지를 전송하여 경로 계산을 요청하게 되는데, 이러한 방식을 SP 탐색(Shortest Path Discovery: 최단거리 경로 탐색) 방식이라고 하고, 그림 3은 SP 알고리즘을 설명한다.

소스노드는 목적지노드까지 경로를 획득하기 위하여 PC-RES-timer를 설정한 후에 IG에게 PC-REQ를 전송한다. PC-REQ를 수신하면, IG는 자신의 TG를 사용하여 경로를 계산한다. 성공하면 해당 노드는 경로를 포함하는 PC-RES를 소스노드에게 응답한다. 그렇지 않으면, 어떤 행위도 취하지 않는다. 소스노드는 PC-RES-timer가 만기가 될 때까지 PC-RES를 받지 못하면 같은 프로세스를 다시 시작한다.

3.2.2. PP (점진적 경로) 탐색 방식

```
At src x that wants to find dst z:
calculate a path from x.TG by using the SP algorithm
IF x fails to find a path THEN
   IF x is not a root THEN
      //to wait for a response
      set PC-RES-timer;
      send PC-REQ to its parent;
      wait for PC-RES until PC-RES-timer expires;
      IF PC-RES-timer expired THEN
         return;
      ENDIF
      IF PC-RES arrives THEN
         initiate packet transmission with the path;
      ENDIF
   ELSE
      return;
   ENDIF
ELSE
    initiate packet transmission with the path;
ENDIF

At a node y that  receives PC-REQ:
Calculate a path from y.TG by using the SP algorithm;
IF y fails  to find a path THEN
   IF y is not a root THEN
      send PC-REQ to its parent;
   ELSE return; ENDIF
ELSE
   send PC-RES with path = (src, …, dst) to src;
ENDIF
```

그림 4. PP(점진적 경로) 탐색 알고리즘
Fig. 4. PP discovery algorithm

SP 탐색방식에서는 PC-REQ 메시지가 항상 IG까지 전달되기 때문에 세션이 많아지면 IG 주위에 혼잡현상이 발생하며 경로 획득 지연시간이 길어진다. 이러한 문제를 부분적으로 해소할 수 있는 방법은 경로

를 탐색하는 과정에서 조상 노드들이 관리하고 있는 부분 토폴로지 정보를 활용하는 것이다. 소스노드에서 IG까지 경로 상에 있는 조상노드들은 이미 부분적인 토폴로지 정보를 가지고 있으며, IG에 가까운 노드일수록 더 많은 토폴로지 정보를 가지고 있다.

경로탐색에서 소스노드에서부터 IG까지 경로계산을 점진적으로 해 나가도록 하는 방식을 PP 탐색 (Progressive Path Discovery: 점진적 경로 탐색) 방식이라고 명하고 그 알고리즘은 그림 4에 설명되어 있다. 소스노드는 먼저 자신의 TG를 사용하여 경로를 찾고, 실패하는 경우에만 PC-REQ를 부모노드에게 전송하여 경로 계산을 요청한다. 부모노드는 경로를 발견하는 즉시 PC-RES를 소스노드에게 보내고, 그렇지 않으면 PC-REQ가 IG에 도달할 때까지 동일한 과정을 반복한다.

3.2.3. 경로 복구

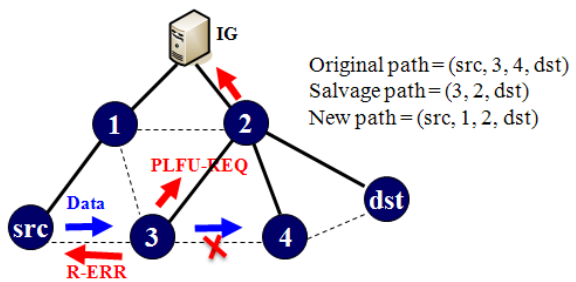

그림 5. 경로복구 과정
Fig. 5. Path recovery process

데이터 전송 시에 어떤 노드가 경로상의 다음노드에게 패킷을 전달하고 ACK를 수신하지 않으면, 대응하는 링크가 파손된 것으로 결정한다. 링크파손을 탐지한 노드는 파손된 링크정보를 포함하는 R-ERR를 소스노드에게 전송하고, 동시에, 보유 중인 패킷을 구제하기 위하여 목적지에 대한 부분경로를 탐색한다. R-ERR을 수신한 소스노드는 남은 패킷을 전송하기 위하여 새롭게 경로획득 프로세스를 시작한다. 탐지노드는 파손된 링크의 정보를 포함하는 PC-REQ를 IG를 향해서 부모노드에게 보낸다. 부분 경로를 포함하고 있는 PC-RES를 받으면, 자신이 가진 패킷을 목적지노드에 보내서 구제한다. PC-REQ를 수신하는 모든 노드는 자신의 TG로부터 파손된 경로 정보를 삭제하고 경로를 계산한다.

그림 5에서, src를 소스노드, dst를 목적지노드로 할 때 원래 경로는 (src, 3, 4, dst)이다. 링크 (3, 4)의 파손을 감지하는 노드 3은 (3, 4)를 포함하는 R-ERR을 src로 보내고, 동시에 (3, 4)를 포함하는 PC-REQ를 부모노드 2에게 보낸다. src는 IG로부터 새로운 경로 (src, 1, 2, dst)를 얻고, 노드 3은 노드 2로부터 구제경로 (3, 2, dst)를 얻는다.

3.3. 루프 문제

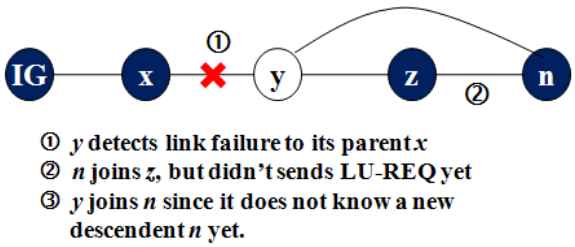

그림 6. 토폴로지 관리에서 루프 형성의 예
Fig. 6. A loop formation example in topology management

트리 관리에서는 오버헤드를 줄이기 위하여 모든 노드가 일정 간격을 두고 변경된 링크상태정보를 갱신하기 때문에 루프가 발생한다. 그림 6는 루프형성의 예를 보여준다. 노드 n이 노드 z에 가입하였지만 노드 y는 노드 n이 후손 노드인지를 아직 알지 못한다고 가정하면 노드 y가 자신의 부모노드 x를 잃은 경우에 자신의 후손노드 n에 가입할 수 있다.

루핑 문제를 효과적으로 해결하기 위하여 트리 깊이 정보와 Hello 메시지를 이용하는 LDR 알고리즘 [20]를 사용한다. 다음 두 조건이 만족되면 해당 트리 노드가 Hello 메시지를 부모노드에게 즉각 보내게 함으로써 루프를 탐지한다.

(1) 트리 노드가 터미널 노드 (leaf)가 아니다; 그리고

(2) 트리노드가 자신의 부모의 HopToIG가 변경되었다는 알게 된다.

Hello 메시지를 엿듣는 노드는 자신의 부모노드의 HopToIG가 변경되었는지를 체크한다. 변경이 된 경우에는 엿듣는 자식 노드는 자신의 HopToIG를 <부모의 HopToIG + 1>로 변경하고, 위의 두 조건이 만족되면 부모노드에게 Hello 메시지를 보낸다. 만일 루프가 존재하는 경우에는 이러한 과정이 계속적으로 반복되고 루프 내에 있는 모든 노드의 HopToIG를 계속적으로 증가하게 된다. 어떤 노드가 자신의 HopToIG가 특정 값, DEPTH-LIMIT를 초과한다는 것을 알면 자신이 루프에 관련되어 있다고 판단하고 자신의 부모를 변경함으로써 루프를 파손한다.

Ⅳ. 성능평가

본 논문에서는 트리 구조를 이용하는 효율적인 이동성 관리와 이동성 관리의 결과로 얻어지는 토폴로

지정보를 이용하는 효율적인 라우팅을 특징으로 하는 TLSR 프로토콜을 제안하였다.

추가적으로 네트워크 오버헤드를 최적화하기 위하여 여러 가지 기법을 사용하였다. 첫째, 이동성 관리 및 경로 탐색에서 플러딩을 사용하지 않는다. 둘째, 제어 메시지의 전달 신뢰도를 높이기 위해서 메시지 전송에서 토폴로지 정보를 이용한 유니케스트를 사용하였으며, Hello 메시지와 같은 브로드케스트 메시지의 경우에는 uBroadcast 방식을 사용하였다. 메시지 전달의 신뢰성을 개선함으로써 메시지의 한 전송 간격 내에 메시지를 수신하지 않으면 해당 링크가 파손되었다고 판단한다. 이것은 또한, 토폴로지 그래프의 정확성을 증가 시킨다. 셋째, 경로 탐색 지연시간을 개선하고 IG 주위의 노드들에게 혼잡현상이 발생하는 것을 줄이기 위하여 PP 탐색 방식을 사용하였다. 넷째, LDR 방식을 적용하여 루프를 빠른 시간에 해결할 수 있는 LDR 방식을 적용하였다. 본 장에서는 TLSR의 성능을 검증하기 위하여 성능지표를 사용하여 평가한다.

4.1. 성능평가지표

먼저 앞에서 논의한 이웃 노드들에게 메시지를 전달하는 Broadcast 방식과 uBroadcast 방식을 비교한다. 다음 두 가지 성능 평가지표를 사용한다.

$$\text{ReceptionRatio(M)} = \frac{\sum_{i \in s.N} \text{nHellosReceived(i)}}{\text{nHellosTransmitted(s)} * |\text{s.N}|}$$

$$\text{ReceptionRatio(UB + Designated)} = \frac{\text{nHellosReceived(D)}}{\text{nHellosTransmitted(s)}}$$

s.N은 노드 s의 이웃 노드의 집합이다. M은 B (Broadcast) 혹은 UB (uBroadcast)이다. i.P는 UB가 사용될 때 Hello 메시지를 수신하는 i의 부모노드를 나타낸다. ReceptionRatio(M)은 기법 M이 사용되는 경우에 소스의 이웃 노드들에서 수신하는 평균 Hello 메시지의 수를 나타낸다. nHellosReceived(x)와 nHellosTransmitted(x)는 노드 x에서 각각 수신하고 전송되는 Hello의 수이다. Designatied는 지정된 수신 노드를 나타낸다.

또한, SP 탐색에 의해서 획득한 경로의 평균 길이와 PP 탐색에 의해서 획득한 경로의 평균 길이를 비교한다. 그리고, 제어 오버헤드에 관한 두 메시지 PC-REQ와 PC-RES의 영향을 분석한다. 마지막으로 제어 오버헤드, 패킷 전송율, 패킷 전송 지연시간, 패킷 전송 지터와 같은 잘 알려진 성능평가지표를 사용한다.

- 제어 오버헤드 (Control Overhead)

$$\text{Control Overhead} = \frac{1}{\text{nNodes}} \sum_{m \in M} \#\text{ of m's transmitted} * \text{size of m}$$

여기서, M = {PC-REQ, PC-RES, R-ERR, Hello, J-REQ, LU-REQ}.

- 평균 전송성공율

각 세션 i에 대하여 DeliveryRatioi를 먼저 다음과 같이 계산한다.

$$DeliveryRatio_i = \frac{\text{nPacketsReceived}(i.d)}{\text{nPacketsSent}(i.s)}$$

여기서, nPacketsReceived(i.d) 와 nPacketsSent(i.s)는 세션 i에 대하여 각각 목적지 d에서 수신한 a패킷의 수와 소스 s에서 송신한 패킷의 수를 나타낸다. 그리고, 평균 전송율을 다음과 같이 나타낸다.

$$\text{Average Delivery Ratio} = \frac{1}{|s|} \times \sum_{i \in s} DeliveryRatio_i$$

- 평균 지연시간

평균지연시간에 대한 평가지표는 다음과 같다.

$$\text{Average Delay} = \frac{1}{k} \times \sum_{i=1}^{k} t_{p_i}^{r} - t_{p_i}^{t}$$

k는 목적지들에서 수신된 전체 패킷을 나타내고, $t_{p_i}^{r}$ 와 $t_{p_i}^{t}$ 는 pi가 수신된 시간 및 전송된 시간을 각각 나타낸다.

- 평균지터

평균 지터를 얻기 위하여, 먼저 각 세션 x에 대한 Jitterx 를 다음과 같이 계산한다.

$$Jitter_x = \frac{1}{k_x - 1} \sum_{i=1,\ldots,k_x-1} |\text{delay}(p_i) - \text{delay}(p_{i+1})|$$

여기서, delay(pi) 는 ith 패킷의 지연시간을 나타낸다. 세션 x에 대하여, 목적지는 소스노드로보터 kx 개의 전송된 데이터 패킷 (p1, p2, ..., pkx)을 수신한다. 그리고, 다음과 같이 평균 지터를 구한다.

$$\text{Average Jitter} = \frac{1}{|S|} \sum_{x \in S} \text{Jitter}_x$$

여기서, |S|는 네트워크에서 실행 중인 전체 세션 수를 나타낸다.

위에 제시한 성능평가지표를 사용하여 TLSR-PP (PP 탐색을 사용하는 TLSR)와 TLSR-PP-LDR (PP 탐색과 LDR을 사용하는 TLSR)을 비교하였으며, PP와 LDR의 효과를 분석하였다. 두 번째 파트에서는 TLSR-PP-LDR과 AODV-Hybrid (라우팅 프로토콜은 AODV를 사용하고 이동성 관리는 Hybrid 방식을 사용) [17]를 비교하였다.

4.2. 시뮬레이션 환경

4.2.1. 시나리오

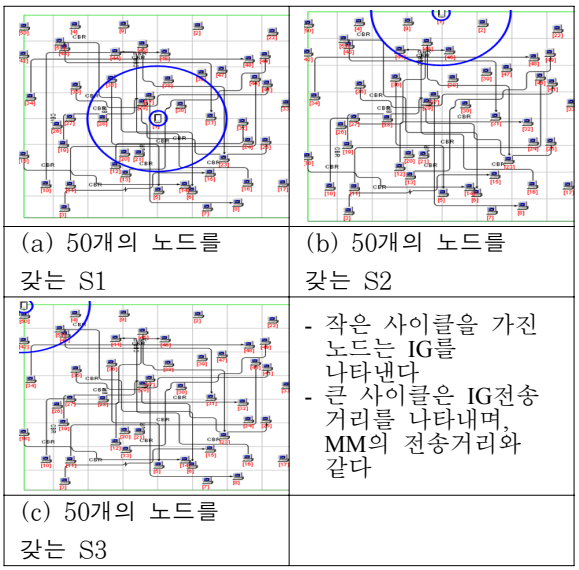

그림 7. 기본 전개 시나리오
Fig. 7. Basic deployment scenario

표 2. 시뮬레이션 파라메타 및 값
Table 2. Simulation parameters and values

Parameter	Value
Mobility Pattern	Random Waypoint
Pause Time	30
nNodes	1 fixed IG, variable nodes
Dimension	1000 x 1000 (m^2)
Transmission range	250 m
Bandwidth	2 Mbps
Traffic pattern	CBR
nSessions and data rate	15 (1 packet/sec) (all sessions are alive until simulation ends)
Packet size	512 bytes
Simulation time	600 seconds

다양한 트리 크기를 고려하기 위하여 그림 7에처럼 IG의 위치에 따른 세 가지 시나리오를 사용하였다. S1, S2, S3는 IG가 중앙, 중앙상단, 코너에 위치하는 전개 시나리오를 나타낸다. QualNet 버전 3.9를 사용하였으며, 표 2는 사용된 파라메타 값을 요약한다.

시뮬레이션은 노드의 수 (nNodes), 노드의 최대속도 (mSpeed), 시나리오 S1, S2, S3를 변경시키면서 수행하였다. Hello 메시지 전송 주기 (Hello_Interval), LU-REQ 메시지 전송 주기 (LU-Interval)는 3초와 5초로 각각 설정되었다. LDR 알고리즘에서 트리의 최대 허용 깊이 (DEPTH_LIMIT)는 6으로 설정하였다. 모든 측정된 값은 서로 다른 Seed를 부여하면서 5번의 실행 결과에 대한 평균값을 취하였다.

4.2.2. 시나리오 평가

표 3. 시나리오 속성 평가
Table 3. Evaluation of key scenario properties

Scenario (S_i)	nNodes	Average number of children	Average number of IG-Subtrees
S1	50	9.5	5.2
	75	14	5.3
	100	17	5.8
S2	50	5.5	8.7
	75	7.8	9.5
	100	11	8.6
S3	50	3.3	15.2
	75	4.8	14.6
	100	5.8	16.3

표 3는 시나리오의 변화에 대하여 IG의 평균 자식노드의 수 및 평균 IG-Subtree의 수를 보여준다. IG-Subtree의 크기는 S1, S2, S3 순서로 증가하고 IG의 자식노드의 수는 반대로 감소한다. 이는 바람직한 변화 패턴을 보여주며, 사용된 시나리오는 프로토콜의 다양한 특징을 평가할 수 있을 정도로 합당하다고 할 수 있다.

4.3. 시뮬레이션 결과

4.3.1. Broadcast와 uBroadcast

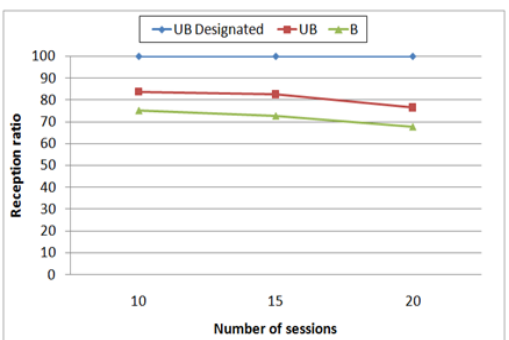

그림 8. 세션 수의 변화에 따른 Broadcast와 uBroadcast의 Hello 수신성공율 (nNodes = 100, mSpeed = 0 m/s)
Fig. 8. Hello reception rate of Broadcast and uBroadcast with the varying number of sessions (nNodes = 100, mSpeed = 0m/s)

어떤 노드가 Broadcast와 uBroadcast를 사용하여 Hello 메시지를 보낼 때 수신노드에서 Hello 메시지의 수신율을 비교하였다. 움직이지 않는 100개의 노드로 구성된 네트워크를 사용하였다. uBroadcast가 사

용될 때, UB Designated는 수신 지정된 노드의 수신율을 나타내고, UB는 일반 엿듣는 노드들의 수신율을 나타낸다. B는 브로드케스트를 사용할 때 이웃 노드들의 수신율을 나타낸다. 그림 8에서처럼, 지정된 노드는 거의 100%를 수신하고 엿듣는 노드들은 거의 80% 이상을 성공적으로 수신한다. 이에 반해 브로드케스트를 사용할 때 이웃 노드들은 약 70%의 메시지를 성공적으로 수신한다.

위 결과에 의하면, Hello 메시지를 전송할 때 자신의 부모노드에게 uBroadcast를 사용하여 전송하는 경우에는 링크가 살아 있다면 거의 100% 전달할 수 있다. 따라서, ACK를 받지 못하면 즉각 부모 링크가 파손되었다고 판단할 수 있다. 반대로 부모 노드는 자식 노드로부터 한 주기 동안 Hello 메시지를 받지 못하면 해당 자식 노드에 대한 링크가 파손되었다고 판단할 수 있다.

4.3.2. 경로탐색 방식에 따른 경로 길이

표 4. 평균 경로 길이 (50개 노드)
Table 4. Average path length (50 nodes)

Secenario / Method	S1	S2	S3
PP discovery	2.27	2.34	2.32
SP discovery	2.23	2.22	2.25

TLSR이 PP와 SP 탐색 방법을 사용할 때 경로의 길이를 비교하였다. 위의 표 4에서 제시하는 바와 같이 PP로 획득한 경로와 SP로 획득한 경로의 길이는 차이가 미미하다는 것을 알 수 있다. PP 탐색 방식은 부분 토폴로지정보를 사용하여 경로를 획득함에도 불구하고 획득한 경로의 길이가 최단거리 경로에 비해서 큰 차이가 없다는 것을 알 수 있다. SP 탐색 방식이 탐색 지연 증가를 유발하고 혼잡현상을 일으킬 수 있다는 것을 고려하면 PP탐색 방식이 적합하다고 판단할 수 있다.

4.3.3. 제어메시지의 오버헤드

이동성 관리 (토폴로지 관리 및 노드 등록)와 라우팅과 관련된 오버헤드를 분석하기 위하여 사용된 여러 가지 메시지들에 대한 데이터를 수집하였다. 표 4와 5에 제시된 것으로써 S3에서 TLSR-PP가 사용될 때 제어메시지의 비정상적인 증가를 보였다. S3에서는 트리 높이가 크기 때문에 고아노드가 자신의 후손에 가입할 가능성이 높다. 따라서, 루프가 발생하였다는 것을 의미한다.

표 5. 시나리오에 따른 라우팅 제어 메시지 분석 (50노드)
Table 5. Analysis of routing control messages according to scenarios

S_i	M*	Routing Overhead: initiated(I) or relayed(R) messages						Total (KB)
		(I) PC-REQ	(R) PC-REQ	(I) PC-RES	(R) PC-RES	(I) R-ERR	(R) R-ERR	
S1	①	1365	569	1278	576	903	1008	2.2
	②	**1365**	**569**	**1278**	**576**	**903**	**1008**	**2.2**
	③	1573	706	1456	732	1100	1226	2.6
S2	①	897	765	783	759	680	792	1.8
	②	**932**	**605**	**845**	**625**	**541**	**600**	**1.6**
	③	1950	2342	1708	2260	1251	1405	4.2
S3	①	1784	7901	1234	910	1238	1392	5.0
	②	**899**	**531**	**810**	**541**	**574**	**654**	**1.5**
	③	1735	3187	1471	3024	1268	1409	4.6

* M(Method): ① TLSR-PP ②TLSR-PP-LDR ③ TLSR-SP-LDR

표 6. 시나리오에 따른 이동성 관리 (토폴로지 관리 및 등록) 제어 메시지 분석 (50 노드)
Table 6. Mobility control (topology management and registration) message analysis according to scenarios (50 nodes)

S_i	M*	Mobility management overhead (topology management + registration): initiated(I) or relayed(R) messages					Total(KB)
		Hello	J-REQ	CR-REQ	(I)LU-REQ	(R)LU-REQ	
S1	①	10477	770	9	6141	2770	4.2
	②	**10847**	**749**	**2**	**6147**	**2714**	**4.2**
	③	10849	769	8	6148	2739	4.4
S2	①	10449	1374	34	6120	8652	6.7
	②	**11016**	**1380**	**45**	**6124**	**8492**	**6.5**
	③	11023	1513	83	6119	8390	6.9
S3	①	10361	2897	447	6037	27808	13.1
	②	**11057**	**1476**	**25**	**6087**	**11404**	**7.7**
	③	10935	1590	86	6090	11316	8.0

* M(Method): ① TLSR-PP ②TLSR-PP-LDR ③ TLSR-SP-LDR

표 5는 시나리오에 따라서 라우팅 제어메시지의 빈도를 나타낸다. TLSR-PP가 사용되는 경우에 (I)PC-REQ의 수는 IG-Subtree의 크기가 감소할 때 감소한다. 이것은 더 큰 트리에 속한 소스노드는 자신의 TG로부터 목적지를 더 잘 얻을 수 있기 때문이다. 하지만, S3에서 TLSR-PP가 사용될 때 (I)PC-REQ의 수의 증가는 경로 획득 메시지가 IG를 향해서 가지 못하도록 방해하는 루프의 발생 때문에 경로 획득이 자주 실패하기 때문이다. S3에서 TLSR-PP가 사용될 때 (R)PC-REQs의 비정상적인 증가는 루프 내에서 반복된 전송 때문이다.

PC-REQ가 항상 IG까지 전송되는 TLSR-SP-LDR을 사용하는 경우에, (R)PC-REQ의 수는 트리의 평균 깊이가 증가하는 S1, S2, S3의 순서로 증가하는 패턴을 보인 것은 당연하다고 할 수 있다. 특히 S3에서는 TLSR-PP-LDR은 TLSR-SP-LDR에 비해서 매우 낮는 오버헤드를 보여준다.

TLSR-PP-LDR은 TLSR-SP-LDR보다 RERR의 생성 수가 훨씬 적다. 그 이유는 TLSR-PP-LSR은 자신의 TG 혹은 현재 획득 하고자 하는 경로 가까운 조상 노드의 TG로부터 경로를 얻을 수 있고, 이들 TG는 IG가 유지하고 있는 TG보다 토폴로지 정보량은 적지만 좀 더 최신의 정보를 가지고 있을 확률이 높기 때문이다. 표 5의 (R)PC-REQ을 보면 PP 탐색방식이 PC-REQ의 전송을 크게 줄인다는 것을 알 수 있다. 표 6에 제시된 이동성관리에 관련된 제어 메시지도 트리 크기의 따라서 유사하게 해석될 수 있다.

4.3.4 TLSR의 성능평가

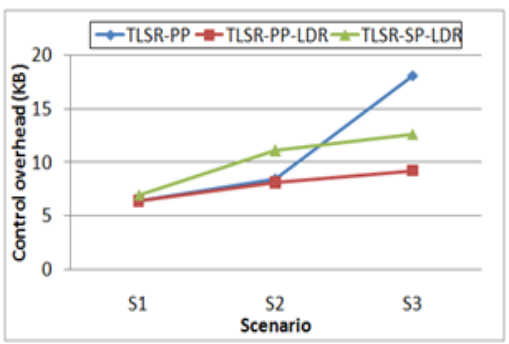

그림 9. 서로 다른 시나리오에 대한 제어 오버헤드 (nNodes = 50, mSpeed =10 m/s)
Fig. 9. Control overhead with different scenarios (nNodes = 50, mSpeed =10 m/s)

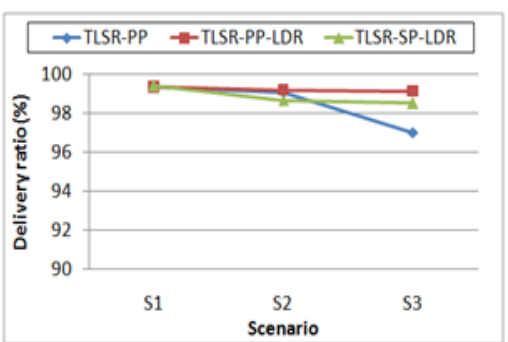

그림 10. 서로 다른 시나리오에 대한 전송성공율 (nNodes = 50, mSpeed = 10 m/s)
Fig. 10. delivery ratio with different scenarios (nNodes = 50, mSpeed =10 m/s)

그림 9-12는 서로 다른 IG 전개 시나리오에 대하여 TLSR의 성능을 평가한다. 그림 9을 보면, S3에서 TLSR-PP는 루프 형성으로 인하여 LDR을 사용하는 방식에 비해 높은 오버헤드를 가진다. 따라서, 그림 13, 14, 15에서처럼, TLSR-PP는 낮은 전송성공율, 높은 지연시간, 높은 지터를 보인다.

또한, 그림 9에서 TLSR-SP-LDR의 오버헤드는 트리의 크기가 증가함에 비례하여 꾸준히 증가한다. 그 이유는 PC-REQ와 PC-RES의 이동거리가 길고, IG가 가진 TG로부터 획득한 경로의 유효성이 떨어지기 때문에 RERR이 더 자주 전송되기 때문이다. 결과적으로, TLSR-SP-LSR은 TLSR-PP-LDR보다 더 나쁜 지연 시간 및 지터를 보여준다. 나머지 논의에서는 LDR 알고리즘을 더 효과적으로 평가하기 위하여 큰 트리를 만드는 시나리오 S3만을 사용한다.

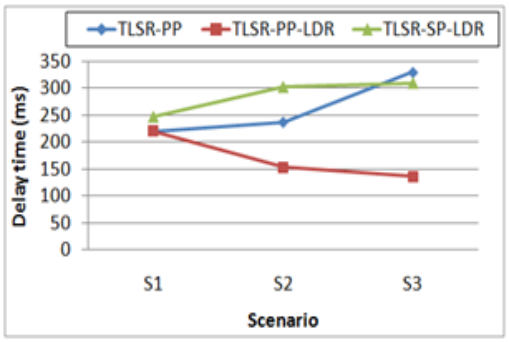

그림 11. 서로 다른 시나리오에 대한 지연시간 (nNodes = 50, mSpeed = 10 m/s)
Fig. 11. Delay with different scenarios (nNodes = 50, mSpeed =10 m/s)

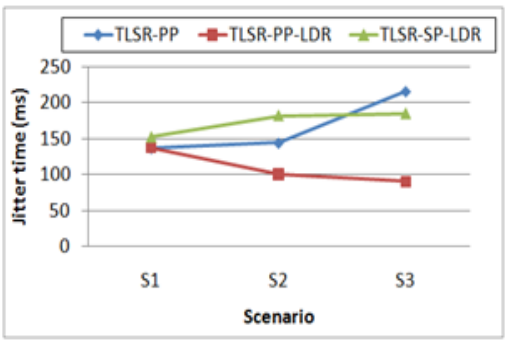

그림 12. 서로 다른 시나리오에 대한 지터 (nNodes = 50, mSpeed = 10 m/s)
Fig. 12. Jitter with different scenarios (nNodes = 50, mSpeed =10 m/s)

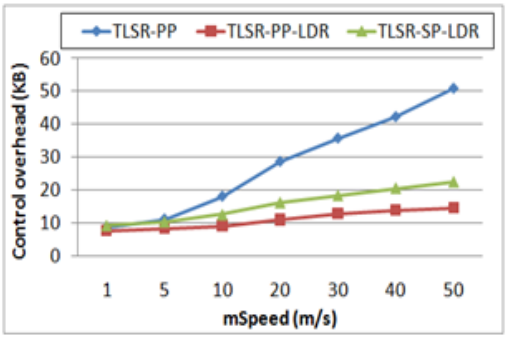

그림 13. mSpeed 변화에 따른 제어오버헤드 (S3, nNodes = 50)
Fig. 13. Control overhead with variation of mSpeed (S3, nNodes = 50)

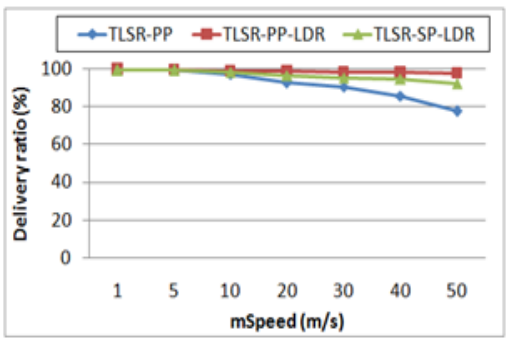

그림 14. mSpeed 변화에 따른 전송성공율 (S3, nNodes = 50)
Fig. 14. Delivery rate with variation of mSpeed (S3, nNodes = 50)

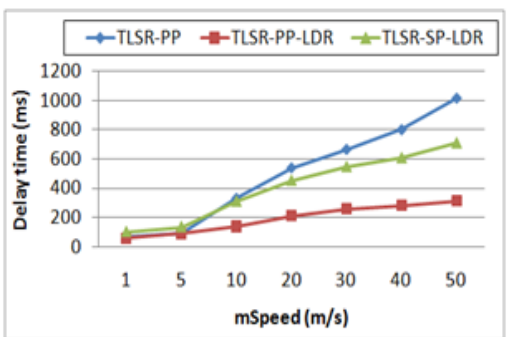

그림 15. mSpeed 변화에 따른 지연시간 (S3, nNodes = 50)
Fig. 15. Delay with variation of mSpeed (S3, nNodes = 50)

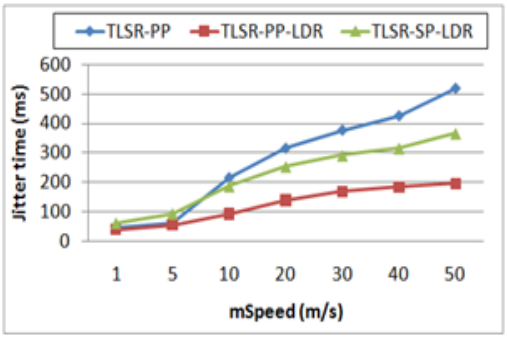

그림 16. nNodes의 변화에 따른 지터 (S3, nNodes = 50)
Fig. 16. Jitter with variation of mSpeed (S3, nNodes = 50)

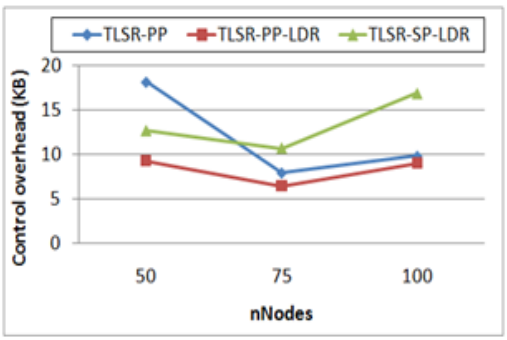

그림 17. 노드 밀도의 변화에 따른 제어 오버헤드 (S3, mSpeed = 10 m/s)
Fig. 17. Control overhead with variation of nNodes (S3, mSpeed = 10 m/s)

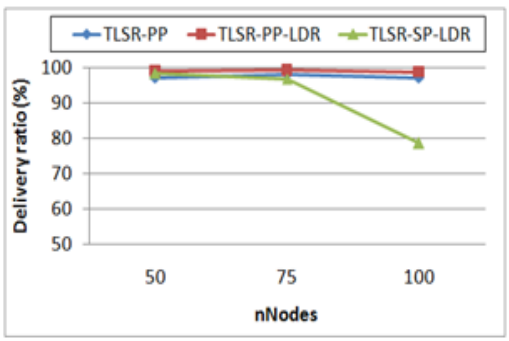

그림 18. 노드 밀도의 변화에 따른 전송성공율 (S3, mSpeed = 10 m/s)
Fig. 18. Delivery rate with variation of nNodes (S3, mSpeed = 10 m/s)

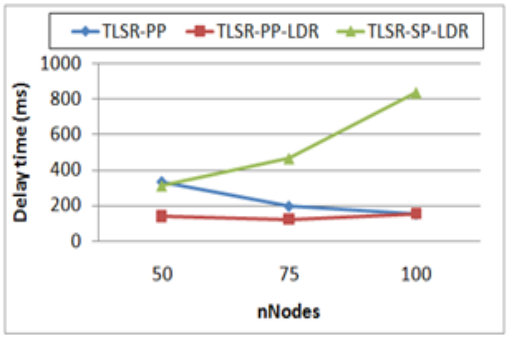

그림 19. 노드 밀도의 변화에 따른 지연시간 (S3, mSpeed = 10 m/s)
Fig. 19. Delay with variation of nNodes (S3, mSpeed = 10 m/s)

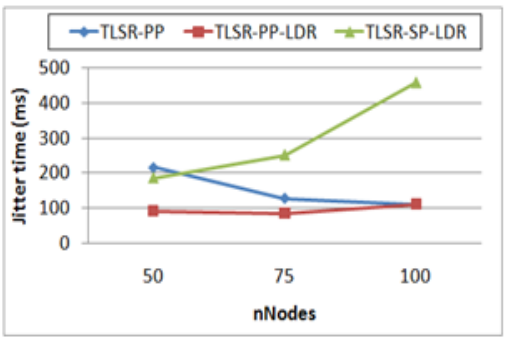

그림 20. 노드 밀도의 변화에 따른 지터 (S3, mSpeed = 10 m/s)
Fig. 20. Jitter with variation of nNodes (S3, mSpeed = 10 m/s)

그림 13-16에서, S3에서 mSpeed를 1m/s에서 초고속인 50m/s로 변경시키고, nNodes를 50으로 고정하여 성능을 평가하였다. 그림 13에서 TLSR-PP에 대하여 제어오버헤드 그래프는 mSpeed가 증가함으로써 급속히 증가하는 패턴을 보여준다. 이것은 노드 이동성이 높으면 더 빈번한 결합 행위가 발생하고, 루프의 형성 빈도를 높인다. TLSR-PP-LDR을 사용하면 모든 성능지표가 안정된다는 것을 알 수 있다. 이것은 패팃 전달율, 지연, 지터가 루프의 형성과 밀접하게 관련되어 있으며, 또한 점증적 경로 탐색의 효과를 제시해 준다. TLSR-PP-LDR은 패킷을 쉽게 구제하기 때문에 전반적으로 패킷 전송성공율이 높으며, 구제경로가 지역적으로 빨리 획득되기 때문에 지연 및 지터도 비교적 느리게 증가한다 (그림 15및 16 참조).

그림 17-20은 노드의 수를 증가시키면서 얻은 결과이다. 시나리오 S3에서 mSpeed = 10m/s로 고정하고, nNodes를 변경시키면서 성능을 평가하였다.

TLSR-PP의 경우에 노드 수가 크게 증가하는 경우에 고아노드가 루프를 형성하지 않고 결합할 수 있는 이웃을 쉽게 얻을 수 있기 때문에 전송성공율이 높다. 동일한 이유로 지연 및 지터도 오히려 약간 감소하는 것을 알 수 있다.

TLSR-SP-LDR은 모든 성능지표에서 가장 큰 변화를 보인다. 그 이유는 PC-REQ는 항상 IG로 전달되고, 노드 수가 많아지면 더 많은 수의 LU-REQ가 IG로 전달된다. 결과적으로, 두 개의 메시지기 IG 주위에 트래픽 혼잡을 야기하고 성능을 떨어뜨린다. 그림 17에서처럼 평균 오버헤드는 그리 높지 않다고 할지라도 IG 주위의 혼잡은 전체적인 성능을 저하시킨다. 하지만, TLSR-PP-LDR은 IG로 전달되는 PC-REQ의 수를 감소시킴으로써 이러한 상황을 누그러뜨린다.

4.3.5. TLSR과 AODV-Hybrid의 비교 평가

TLSR-PP-LDR와 AODV-Hybrid[17]를 비교 평가하였다. Hybrid 이동성 관리 방식은 다음과 같이 구현되었다, IG는 k 홉까지 주기적으로 광고 메시지를 플러딩하고, 수신 노드들은 IG까지 역방향 경로를 설정한다. 광고 메시지를 수신하면 등록되지 않은 노드는 역방향 경로를 이용하여 등록하고 IG로부터 자신의 노드까지 순방향 경로를 설정한다. 각 경로는 path-timer를 설정하고, 이 타이머가 만기되면 경로는 무효로 된다. 광고메시지를 수신하지 못하는 노드, 즉 등록하지 못한 노드는 Expanding Ring Search 방법[17]을 사용하여 기 등록된 노드 혹은 IG를 찾기 위하여 응답요청 (Solicitation) 메시지 (TTL = 2)를 플러딩한다. 이 과정에서 해당 노드는 이 메시지를 수신하는 기 등록된 노드까지 순방향 경로를 설정하게 된다. 응답요청 메시지를 수신하면, 등록된 메시지는 응답메시지를 등록을 원하는 요청노드에게 보낸다. 등록을 원하는 노드는 응답한 기 등록된 노드까지 역방향 경로를 형성하게 된다. 결국, 등록을 원하는 노드는 두 역방향 경로를 연속적으로 거치면서 IG에 등록을 할 수 있다. 파라메타 값들은 복수의 실행을 통해서 적합한 값으로 선택되었으며, 표 7에 요약 하였다.

표 8을 보면, TLSR-PP-LDR의 경우에 라우팅 오버헤드가 매우 낮음을 알 수 있다. 이는 본 프로토콜이 의도한 바와 같이 라우팅 프로토콜이 이동성관리 프로토콜의 결과로 얻어지는 토폴로지 그래프를 활용하기 때문이다. 이동성 관리의 효율성(MME)과 라우팅 효율성(RTE)면에서 TLSR-PP-LDR은 AODV-Hybrid보다 훨씬 우수하다는 것을 알 수 있다. 또한, 표 8은 플러딩이 오버헤드를 크게 증가시키고 전반적인 성능을 떨어뜨린다는 것을 입증한다.

표 7. AODV-Hybrid를 위한 파라메타 값
Table 7. Parmeter values for AODV-Hybrid

Parameter	value	remarks
Transmission hop count of registration message	K=2	K = TTL
Transmission hop count of Response request message	K=2	K increases by 2 if a node fails to find a registered node or an IG
Transmission interval of response request message	5 sec	
Advertize message interval	2 sec	
RREQ transmission hop count	TTL=2	TTL increases by 2 if a node fails to find IG or destination

표 8. 라우팅 및 등록 오버헤드 (50 nodes)
Table 8. Routing and Registration overhead (50 noes)

S_i	Protocol	Routing overhead (KB)	Registration overhead (KB)
S1	TLSR-PP-LDR	2.2	4.9
	AODV-Hybrid	11.9	20.0
S2	TLSR-PP-LDR	1.6	7.3
	AODV-Hybrid	22.4	67.1
S3	TLSR-PP-LDR	1.5	8.6
	AODV-Hybrid	26.4	83.7

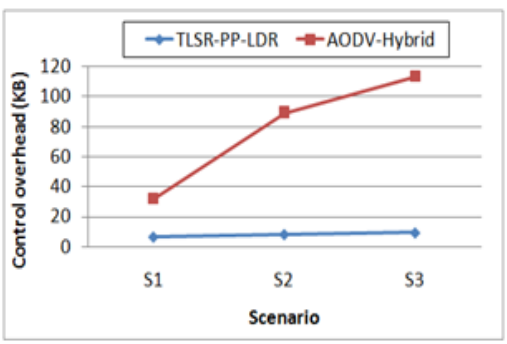

그림 21. 서로 다른 시나리오에 대한 제어 오버헤드 (nNodes = 50, mSpeed = 10 m/s)
Fig. 21. Control overhead with different scenarios (nNodes = 50, mSpeed = 10 m/s)

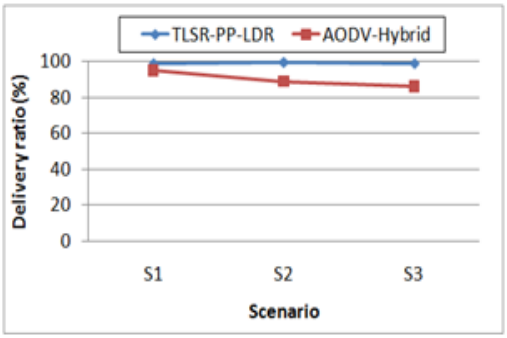

그림 22. 서로 다른 시나리오에 대한 전송성공율 (nNodes = 50, mSpeed = 10 m/s)
Fig. 22. Delivery rate with different scenarios (nNodes = 50, mSpeed = 10 m/s)

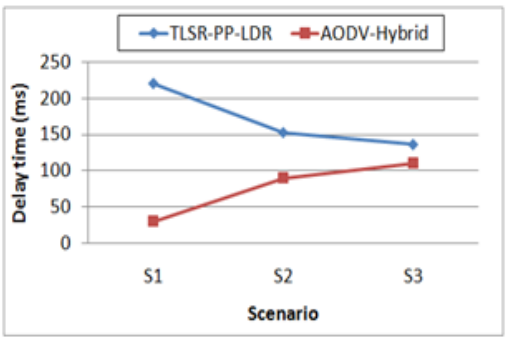

Fig. 23. 서로다른 시나리오에 대한 전송 지연시간 (nNodes = 50, mSpeed = 10 m/s)
Fig. 23. Delay with different scenarios (nNodes = 50, mSpeed = 10 m/s)

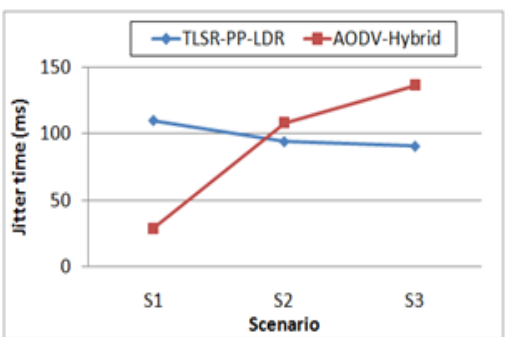

Fig. 24. 서로 다른 시나리오에 대한 지터(nNodes = 50, mSpeed = 10 m/s)
Fig. 24. Jitter with different scenarios (nNodes = 50, mSpeed = 10 m/s)

그림 21에서, TLSR-PP-LDR과 AODV-Hybrid 사이의 갭이 S1, S2, S3로 바뀌면서 증가하는 것은 트리의 높이, 즉 IG와 MN 사이의 평균적인 거리에 비례하여 플러딩이 더 부정적인 효과를 보인하다는 것을 나타낸다. AODV의 전송성공율은 오버헤드의 증가에 비례하여 감소한다는 것을 알 수 있다 (그림 22). 표 8를 보면, TLSR-PP-LDR의 라우팅 오버헤드는 트리

가 증가함으로써 오히려 약간 감소한다는 것을 보인다. 이것은 트리 사이즈가 크면 소스노드 주위에서 경로를 얻게 될 확률이 높고, 그에 따라서 PC-REQ의 재전송 수가 오히려 줄어들게 된다 (표 5의 (R)PC-REQ 참조). 이것이 TLSR-PP-LDR의 지연 커브와 지터 커브가 트리의 크기가 증가함에 따라서 감소하는 이유라고 할 수 있다 (그림 23-24).

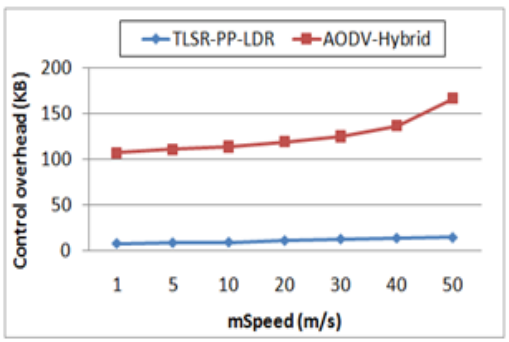

그림 25. mSpeed 변경에 따른 제어오버헤드 (S3, nNodes = 50)
Fig. 25. Control overhead with variation of mSpeed (S3, nNodes = 50)

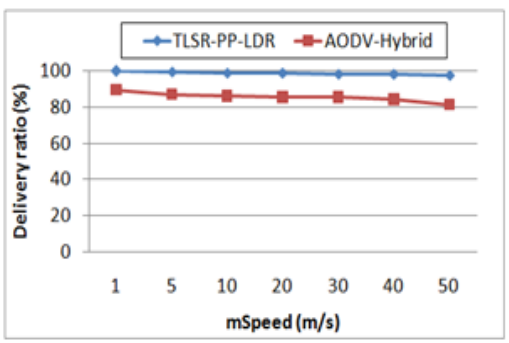

그림 26. mSpeed 변경에 따른 전송성공율 (S3, nNodes = 50)
Fig. 26. Delivery rate with variation of mSpeed (S3, nNodes = 50)

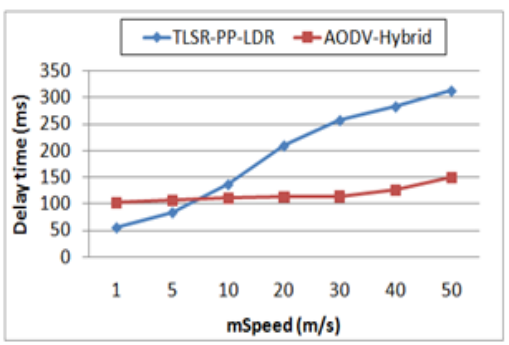

그림 27. mSpeed 변경에 따른 전송지연시간 (S3, nNodes = 50)
Fig. 27. Delay with variation of mSpeed (S3, nNodes = 50)

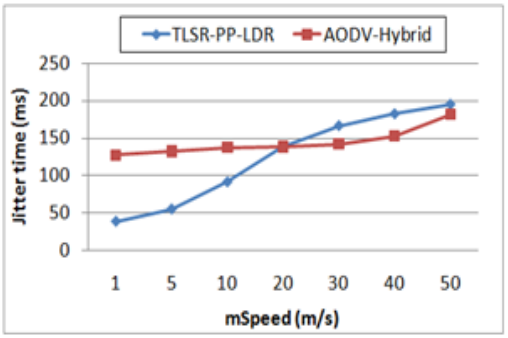

그림 28. mSpeed변경에 따른 지터 (S3, nNodes = 50)
Fig. 28. Jitter with variation of mSpeed (S3, nNodes = 50)

S3에서 nNodes를 50으로 고정하고 mSpeed를 변경시키면서 두 프로토콜을 비교 하였다 (그림 25-28). 그림 25는 두 프로토콜 사이의 높은 오버헤드 갭을 보여준다. 두 프로토콜의 제어 오버헤드는 속도 증가에 따라 파손 링크가 증가하기 때문에 증가한다. TLSR-PP_LDR의 오버헤드도 속도가 증가함에 따라서 2배 이상 증가하지만 전반적으로 오버헤드가 낮기 때문에 그래프 상에 표시가 크게 나지 않는다. 이에 반하여, AODV-Hybrid는 속도에 덜 민감하지만 속도가 매우 높으면 (40m/s 이상)빠르게 증가함을 알 수 있다.

TLSR-PP-LDR은 매우 우수한 전송성공율을 보여주지만 지연 및 지터 커버는 지속적으로 증가하는 패턴을 보여준다. 이것은 더 높은 이동성이 TG를 더 빨리 낡게 만들고 새로운 경로에 대하여 더 자주 탐색하게 만들기 때문이다. 지터는 지연 커버와 유사한 커버 패턴을 보인다. 이것은 패킷을 전송 중에 경로가 파손되는 횟수가 증가하고, 경로 재탐색 후에 구제되는 패킷은 전송 지연 시간이 길기 때문에 연속되는 전송 패킷들 사이의 지연 편차가 커지기 때문이다.

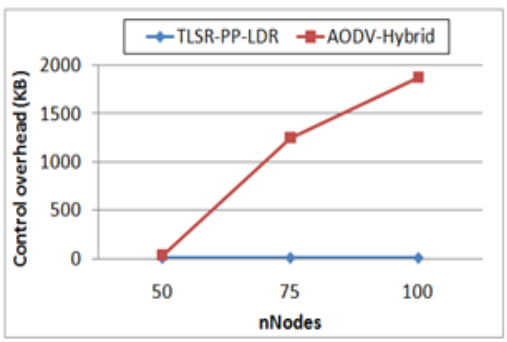

그림 29. 노드밀도 변화에 따른 제어오버헤드 (S1, mSpeed = 10 m/s)
Fig. 29. Control overhead with variation of nNodes (S1, mSpeed = 10 m/s)

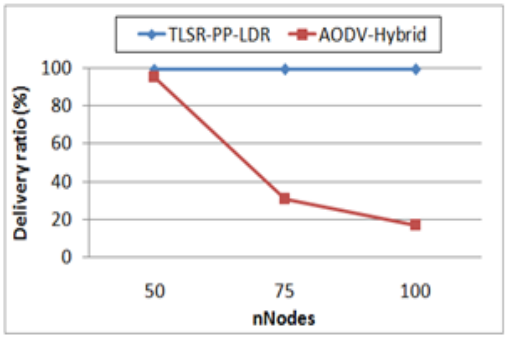

그림 30. 노드밀도 변화에 따른 전송성공율 (S1, mSpeed = 10 m/s)
Fig. 30. Delivery rate with variation of nNodes (S1, mSpeed = 10 m/s)

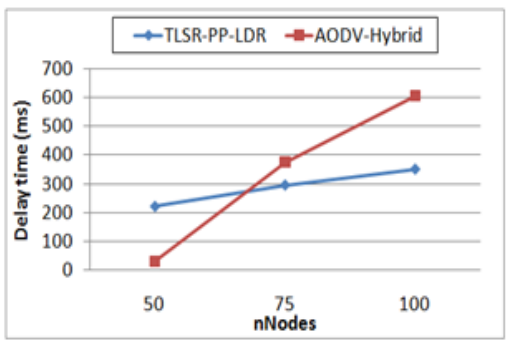

그림 31. 노드밀도 변화에 따른 지연시간 (S1, mSpeed = 10 m/s)
Fig. 31. Delay with variation of nNodes (S1, mSpeed = 10 m/s)

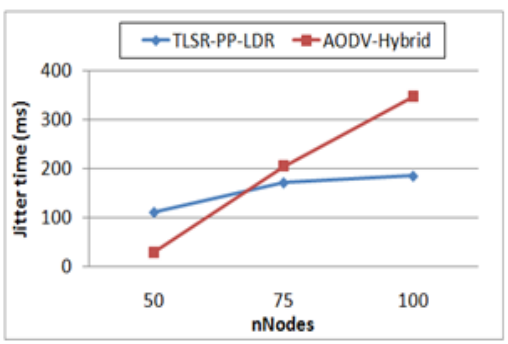

Fig. 32. 노드밀도 변화에 따른 지터 (S1, mSpeed = 10 m/s)
Fig. 32. Jitter with variation of nNodes (S1, mSpeed = 10 m/s)

시나리오 S1을 사용하여 nSpeed를10m/s로 고정하고 nNodes를 변경시키면서 프로토콜을 비교하였다 (그림 29-32). AODV-Hybrid는 노드 수가 증가하면 성능이 크게 떨어진다는 것을 알 수 있다. S2와 S3가 사용되면 이러한 결과는 더 악화된다는 것을 확인하였다. TLSR-PP-LDR은 지연시간 및 지터가 노드 수의 증가에 따라서 약간 증가하지만 AODV-Hybrid와 비교하면 높은 밀도에서 조차도 매우 안정적이라는 것을 알 수 있다.

V. 결 론

본 논문에서는 트리 기반의 이동성 관리 방법과 경로 탐색 알고리즘을 포함하는 새로운 TLSR 프로토콜을 제안하였다. 이동성 관리 과정에서 부수적으로 얻어지는 토폴로지 그래프를 활용하기 때문에 경로 탐색 알고리즘이 적은 오버헤드를 가지고 효율적으로 작동한다는 것을 확인 하였다. 또한 IG주위에 트래픽 혼잡현상이 성능에 부정적인 영향을 끼친다는 것을 확인하였으며, 이를 해결하기위한 방법으로 점진적인 경로탐색 방식이 제시되었다. 또한, 플러딩을 완전히 배제하였고, 제어메시지의 전송 신뢰성을 높이기 위하여 유니케스트 기반의 브로드케스트를 사용하였다. 이 방식은 링크의 유효상태를 빠르게 판단할 수 있도록 해 주기 때문에 토폴로지 그래프의 정확성을 크게 개선한다. 또한, 제어 오버헤드를 줄여서 제어 메시지 전송을 최적화하는 기법을 제시하였다. 결과적으로, TLSR은 전반적으로 아주 낮은 제어 오버헤드를 유지하였고, AODV-Hybrid에 비해 이동성이 높고 노드수가 크게 증가하여도 우수한 성능 보였으며, 확정성이 뛰어나다는 것을 입증하였다.

참 고 문 헌

[1] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad Hoc Networks*, vol. 17, no.8, pp. 1369-1379, Aug. 1999.

[2] C. Perkins, "IP Mobility Support," *Request For Comments (Standard) 2002*, Internet Engineering Task Force, Oct. 1996.

[3] 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.

[4] 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.

[5] D. B. John and D. A. Malz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, edited by T. Imielinski and H. Korth, Kluwer Academic Publishers, ch.5, pp. 153-181, 1996

[6] D. Jonsson, F. Alriksson, T. Larsson, P. Johansson and J. G. Maguire, "MIPMANET - mobile IP for mobile ad hoc networks," in *Proceedings of IEEE/ACM Workshop on Mobile and Ad Hoc Networking and Computing (MobiHoc'00)*, pp. 75 – 85, Aug. 2000.

[7] H. Ammari and H. El-Rewini, "Integration of mobile ad hoc networks and the Internet using mobile gateways," in *Proceedings of 18th International Symposium of Parallel and Distributed Processing*, vol. 13, pp. 218b, Apr. 2004.

[8] H. El-Moshrify, M.A. Mangoud, and M. Rizk, "Gateway discovery in ad hoc on demand distance vector (AODV) routing for Internet connectivity," in *Radio Science Conference*, NRSC 2007, pp. 1-8, Mar. 2007.

[9] H. Oh and S.Y. Yun, "Proactive cluster-based distance vector (PCDV) routing protocol for mobile ad hoc networks," *IEICE Transactions on Communications*, vol. E90-B, no.6 , pp.1390-1399, Jun. 1, 2007.

[10] H. Oh, "A tree-based approach for the Internet connectivity of mobile ad hoc networks," *Journal of Communications and Networks*, vol. 11, no. 3, pp. 261-270, Jun. 2009.

[11] J. Broch, D. A. Maltz and D. B. Johnson, "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks," in *Proc. Int'l. Symp. Parallel Architecture, Algorithms, and Networks*, Perth, Australia, pp. 370-375, Jun. 1999.

[12] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 21-32, Jan. 1987.

[13] J. McQuillan, I. Richer, and E. Rosen, "The new routing algorithm for the ARPANET," *IEEE Transaction on Communications*, vol. 28, no. 5, pp. 711,719, May. 1980.

[14] M. Caleffi, G. Ferraiuolo, L. Paura, "Augmented tree-based routing protocol for scalable ad hoc networks," in the *Proceedings of the IEEE Internatonal Conference Mobile Adhoc and Sensor Systems,* 2007. MASS 2007. pp. 1-6, Oct. 2007.

[15] P. Jacquet, P. Muhlethaler, and A. Qayyum, "Optimized link state routing protocol," *IETF MANET*, Internet Draft, Nov. 1998.

[16] P. Ruiz, A. Gomez-Skarmeta, "Enhanced internet connectivity for hybrid ad hoc networks through adaptive gateway discovery," in Proceedings of *the 29th Annual IEEE International Conference on Local Computer Networks* (LCN'04), pp. 370 – 377, Nov. 2004.

[17] P. Ratanchandani, and R. Kravets, "A hybrid approach to Internet connectivity for mobile ad hoc networks," in *Proceeding of IEEE WCNC* 2003, pp. 1522 – 1527, Mar. 2003.

[18] Y.-C. Tseng, C.-C. Shen, and W.-T. Chen "Integrating mobile IP with ad hoc networks," *IEEE Computer*, vol. 36, no. 5, May 2003.

[19] Y. Sun, E. M. Belding-Royer, and C. E. Perkins, "Internet connectivity for ad hoc mobile networks," *International Journal of Wireless Information Networks special issue on Mobile Ad Hoc Networks (MANETs): Standards, Research, and Applications*, vol. 9, no. 2, pp. 75-88, Apr. 2002.

[20] T-D Han, H. Oh, "Detecting and resolving a loop in the tree-based mobility management protocol," *LNCS 6104*, pp. 583-592, May 2010.

[21] W. Peng, Z. Li, F. Haddix, "A practical spanning tree based MANET routing algorithm," in: *the Proceedings of the 14th International Conference Computer Communications and Networks*, 2005. ICCCN 2005, pp. 19-24, Oct. 2005.

[22] S.W. Lee, C.T. Ngo, T.D. Han, J.W. Kim, H. Oh, "Resolving the funneling effect in the node mobility management of infrastructure-based mobile ad hoc networks)," The Journal of Korea Information and Communications Society, vol. 36, no. 12 pp. 984-993, Dec. 2011

이 성 욱 (Sung Uk Lee)

1998년 경북대학교 전자공학 학사
1998년~2001년 대우조선해양
2004년 University of Florida 전기컴퓨터 석사
2010년 University of Florida 전기컴퓨터 박사
2010~현재 포항산업과학연구원 선임연구원
<관심분야> 이동통신, 무선 애드혹 및 센서 네트워크

오 지 충 (Chi-Trung Ngo)

2010년 베트남 호치민 공과대학 공학사
2010년~현재 울산대학교 전기공학부 석사과정
<관심분야> 무선통신, 상황인지 컴퓨팅

한 충 진 (Trung-Dinh Han)

1999년 베트남 호치민 공과대학 공학사
2002년 베트남 호치민 공과대학 석사
2011년 울산대학교 컴퓨터정보통신공학 박사
2011년~현재 호치민산업대학 연구교수
<관심분야> 무선센서네트워크, 이동 애드혹 네트워크, 임베디드시스템, 디지털이미지처리

김 제 욱 (Je-Wook Kim)

2004년 울산대학교 공학사
2007년 울산대학교 정보통신공학석사
2010년~현재 울산대학교 자동차선박대학원 박사과정
<관심분야> 상황인지컴퓨팅, 무선통신

오 훈 (Hoon Oh)

1981년 성균관대학교 전자공학 학사
1993년 텍사스A&M대학교 전산학 석사
1995년 텍사스A&M대학교 전산학 박사
1996년 삼성전자 중앙연구소 수석연구원
2005년~현재 울산대학교 컴퓨터정보통신공학전공 부교수
<관심분야> 실시간시스템, 임베디드시스템, 상황인지컴퓨팅, 무선 네트워크 프로토콜