논문 15-40-07-12 The Journal of Korean Institute of Communications and Information Sciences '15-07 Vol.40 No.07
http://dx.doi.org/10.7840/kics.2015.40.7.1313

MANET환경에서 Interrupt Message와 Backup path 기법에 기반한 AODV의 성능개선

이 윤 경*, 김 주 균°

Performance Enhancement of AODV Routing Protocol Based on Interrupt Message and Backup Path Strategy in MANET

Yun-kyung Lee*, Ju-gyun Kim°

요 약

MANET 환경에서 빈번한 경로 단절은 반복된 경로 탐색으로 인해 제어 패킷 오버헤드와 패킷 손실을 증가시킨다. AODV-I는 event driven방식을 이용하여 주기적인 Hello 메시지를 제거함으로써 AODV의 성능을 개선하였다. Hello 메시지와 달리 각 이벤트마다 전송되는 인터럽트 메시지는 이웃노드의 상태를 알 수 있어 링크 단절을 탐지 및 예측도 할 수 있다. 따라서 AODV-I의 성능은 각 인터럽트 메시지 종류에 따라 처리 절차를 추가함으로써 더욱 개선될 수 있고, 또한 AODV의 단일경로로 인한 문제를 백업 경로기법을 추가하여 개선하는 것도 가능하다. 본 연구에서는 경로 탐색 수와 전송 지연을 줄이기 위해 개선된 백업 경로방식과 인터럽트 메시지 방식을 결합한 AODV-IB를 제안한다. AODV-IB는 각 인터럽트 메시지마다 링크 단절 예측과 탐지에 대한 처리 절차를 추가하여 AODV-I를 개선한다. 또한 추가적인 제어 패킷 없이 지연을 최소화하도록 개선한 백업 경로 방식도 구현한다. QualNet 5.0을 사용하여 구현한 시뮬레이션 결과는 AODV-I에 비해 제안하는 AODV-IB가 더 좋은 성능을 나타낸다.

Key Words : MANET, AODV, AODV-BR, Interrupt message, Backup path

ABSTRACT

In MANET, frequent route breaks lead to repeated route discovery process and this increases control packet overhead and packet drop. AODV-I improves performance of AODV by using the event driven approach which removes periodic Hello message. Unlike the Hello message, Interrupt message which is sent for each event can detect and predict the link failure because it allows node to know the status of the neighbor node. From this characteristics of Interrupt message, performance of AODV-I can be further improved by adding a processing procedures for each type of Interrupt message and it is also possible to improve AODV-I by adding the Backup path scheme because it originally has problems due to a single path of AODV. In this paper, we propose AODV-IB that combines improved Backup path scheme and Interrupt message approach of AODV-I in order to reduce transmission delay and the number of route discoveries. AODV-IB improves AODV-I by adding proper processing procedures for the link failure prediction and detection for each Interrupt message. We also implement improved Backup path strategy in AODV-IB by minimizing delay without additional Control packet. Simulation results, using the simulator QualNet 5.0, indicate that proposed AODV-IB performs better than AODV-I.

※ This Research was supported by the Sookmyung Women's University Research Grants (1－1403 - 0051)
◆ First Author : Sookmyung Women's University Department of Computer Science, lyk4707@sookmyung.ac.kr, 학생회원
° Corresponding Author : Sookmyung Women's University Department of Computer Science, jgkim@sookmyung.ac.kr, 정회원
논문번호 : KICS2015-04-140, Received April 30, 2015; Revised July 24, 2015; Accepted July 24, 2015

Ⅰ. 서 론

이동 애드혹 네트워크(MANET) 환경에서 모든 노드들은 이동성을 가지고 있기 때문에 시간이 지남에 따라 네트워크 토폴로지가 동적으로 변화하고, 배터리 상태에 따른 데이터 전송반경이 제한되며, 이 외에도 무선 환경의 특성들 즉, 간섭(Interference), 멀티패스 페이딩(Multipath fading), IEEE 802.11의 contention 특성으로 인한 충돌(Collision), 각 이동 노드에서의 큐 오버플로우(Queue overflow) 등으로 인하여 링크 단절이 빈번하게 일어나는 불안정한 라우팅 경로를 갖는다. 따라서 이러한 빈번한 링크 단절에 대해 라우팅 프로토콜의 민첩하고 적절한 대응이 필요하다[1-2].

링크 단절이 발생하는 원인은 크게 두 가지로 나눌 수 있는데, 배터리 용량의 부족이나 노드의 불규칙적인 이동에 의해 경로상의 next node가 더 이상 전송범위에 없어 경로 재탐색을 해야 하는 단절과 무선 환경의 특성으로 인해 발생하는 일시적인 단절이다. 후자의 경우는 링크 단절이 발생하여도 next node는 여전히 전송 범위 안에 존재하지만, 기존 AODV(Ad hoc On-demand Distance Vector)에서는 이 두 가지 경우 모두 경로 복구를 수행하는 비효율적인 부분이 있다[1]. 이러한 잦은 링크 단절 문제를 완화시키기 위해 AODV는 Hello 메시지를 정의하고, 경로상의 이동 노드들이 서로 비콘(Beacon)으로 이용할 수 있도록 하였지만 여전히 링크 단절은 많이 일어나며, 오히려 주기적인 Hello 메시지는 네트워크 혼잡의 원인이 되어 여러 가지 문제점을 발생시킨다[1,4].

[3]에서 밝힌 AODV의 성능에 영향을 미치는 다양한 factor들과 성능 메트릭들 간의 상관관계 분석에서 Hello 메시지의 주기와 성능 분석을 기반으로, [4]에서는 주기적인 Hello 메시지를 event driven 방식으로 대체하여 성능을 개선한 AODV(AODV using Interrupt message 이하 AODV-I라고 함)를 제안하였다. 즉, 요구기반 방식의 라우팅 프로토콜인 AODV에서 "Reactive"하지 않아 논쟁의 근원이 되고, 네트워크 혼잡으로 인해 여러 가지 문제점을 발생시키는 주기적인 Hello 메시지를 AODV-I에서는 로컬 연결에 영향을 주는 이벤트에 따라 미리 정의된 인터럽트 메시지를 전송하는 것으로 대체하여 라우팅 오버헤드와 종단간 지연, 그리고 네트워크 혼잡을 줄이고, 패킷 전달률과 네트워크 수명을 향상시키는 등 성능을 개선하였다[4].

AODV-I에서 정의한 인터럽트 메시지들은 Hello 메시지와 달리 그 종류에 따라 이웃 노드의 상태를 알 수 있어 링크 단절을 예측 또는 탐지가 가능하기 때문에 각 경우에 따라 처리 절차를 추가하여 성능이 더욱 개선될 수 있다.

또한 AODV-I는 백업 경로를 사용하지 않음으로서 단일 경로만을 갖는 AODV의 문제점을 그대로 가지고 있어 링크 단절로부터 효율적인 경로 복구를 제공하는 백업 경로 기법을 통한 성능 개선도 가능하다[4].

AODV의 백업 경로에 관한 선행연구들 중 AODV-BR(Backup Route)은 경로 탐색 과정에서 RREP(Route REPly)의 엿듣기(Overhearing)를 통해 백업 경로에 관한 정보를 비교적 간단히 확보하게 되고, 링크 단절 시 주경로(Primary path)에 지리적으로 근접한 백업 노드로 우회(Bypass)할 수 있어 빠른 경로 복구와 적정한 라우팅 길이를 제공하는 장점이 있다. 하지만 네트워크 밀도가 높아지는 경우에는 참여하는 백업 노드들이 많아지면서 라우팅 오버헤드가 높아지게 되고, 특히 링크 단절 시 AODV-BR은 next node를 선택하지 않고 모든 백업 노드에게 데이터 패킷을 브로드캐스트하여 데이터 중복 문제를 발생시킨다[5]. 이 문제를 개선하려는 연구들에서는 링크 단절 시 백업 노드들 가운데 next node를 선정하기 위해 추가적인 제어패킷을 사용하거나 distance, energy 등의 특정 metric을 통해 next node를 선택하는 방법을 제안하고 있지만, 추가적인 라우팅 오버헤드와 전송 지연을 발생시키는 제한점은 여전하기에 개선이 필요하다[6-12].

이에 본 연구에서 제안하는 AODV-IB(AODV based on Interrupt message and Backup path strategy)는 AODV-I에서 정의한 인터럽트 메시지들을 이용하여 동적인 위상 변화를 반영한 이웃 목록과 로컬 연결 정보를 유지하고, 수신된 인터럽트 메시지의 종류에 따라 처리 절차를 추가하여 개선함으로써, 데이터 전송 시 링크 단절이 예측되거나 탐지된 경우 AODV-BR의 문제점을 보완한 백업 경로 방식을 이용하여 경로 복구를 한다.

또한 AODV-IB에 적용한 개선된 백업 경로 방식은 추가적인 제어패킷 없이 빠르게 경로 복구를 수행하여, 경로 재탐색 빈도를 줄임으로써 전송 지연과 라우팅 오버헤드를 최소화하여 성능을 개선한다.

아울러 [9]는 네트워크에서 발생되는 실패(Failure)를 갑작스런 물리적 고장이나 무선 환경의 단절에 의한 예측 불가능한 실패(Unpredictable failure)와 노드의 이동 혹은 부족한 전력 등에 의한 예측 가능한 실패(Predictable failure)로 분류하였는데, AODV-I에서는 예측 가능한 실패를 연구 범위로 한정한 반면 제안

하는 AODV-IB에서는 두 가지 실패 모두를 연구 대상으로 하였다.

본 논문에서 제안하는 AODV-IB의 성능을 실험 분석하기 위해 QualNet 5.0을 이용하였고, 시뮬레이션 파라메터들과 성능 메트릭들은 [3]에서 제안한 AODV 라우팅 프로토콜의 성능 메트릭들 간의 상관관계 분석을 기반으로 선정하였으며, 각 파라메터들의 변화에 따라 성능 메트릭들을 기준으로 선행 연구 AODV-I와 제안하는 AODV-IB의 성능을 비교 분석하였다.

본 논문의 구성은 다음과 같다. II. 관련 연구에서는 AODV의 백업 경로에 관한 선행 연구들을 정리하고, III.에서는 우선 인터럽트 메시지 방식을 제안한 선행 연구 AODV-I와 본 연구에서 제안하는 개선된 백업 경로 방식에 관해 설명하고, 이를 기반으로 제안하는 AODV-IB에 대해 기술하였다. IV. 실험 분석에서는 AODV-I와 제안하는 AODV-IB의 성능을 비교 분석하기 위한 시뮬레이션 환경 설정과 그 실험 결과를 분석하여 기술하였으며, V.에서 결론을 맺는다.

Ⅱ. 관련 연구

AODV의 백업 경로에 관한 선행 연구는 링크 성능 평가 척도를 이용하여 주 경로와 대체 경로(Alternate path)를 구성하는 연구들과 RREP를 엿들어 간단히 백업 경로를 구성하는 AODV-BR 계열의 연구들로 분류할 수 있다. 전자에서 사용되는 링크 성능 평가 척도로는 최단거리, 잔존 에너지량, 신호의 세기, Congestion (Queue status value), Hello 메시지 수신률, LET(Link Expire Time), Link/Node disjoint path 등이 있고, 이를 통해 주경로와 대체 경로를 선정하는 연구들로는 [13-21]가 있다. 링크 성능을 고려한 신뢰성 있는 경로의 채택으로 데이터 전송률을 높이고, 경로 재탐색 빈도를 줄여 성능을 개선한 반면, 성능 평가를 위한 추가적인 지연이나 제어 패킷 오버헤드가 발생되는 것과 구현상의 복잡함, 동적 위상변화에 대한 제한점이 문제점으로 지적되고 있다.

후자의 경우 우선 AODV-BR은 경로 탐색과정에서 주경로를 위한 RREP를 엿들어 백업 경로를 생성하고, 링크 단절 시 모든 백업 노드에게 데이터 패킷을 브로드캐스트하여 패킷 전달률과 전송 지연을 개선하였으나, 중복 데이터로 인한 성능 저하 문제가 지적되고 있다. 이 문제점을 개선하려는 연구들로 AODV-ABR (AODV-Adaptive Backup routing)[6]에서는 링크 단절 시 BRRQ(Backup Route ReQuest), BRRP (Backup Route REply)를 이용하여 3-way handshake를 통한 복구를 제안하였지만 추가된 제어 패킷으로 오버헤드가 크며, AODV-RD (AODV based on Reliable Delivery)[7]에서는 링크 단절 예측 기법을 제안하고, 링크 단절 시 Communicating power가 강한 alternate node를 선택하도록 하였으나 동적 위상 변화에서의 제한점은 여전하다. SOMR(Stable On-demand Multipath Routing)[8]에서는 RREP를 엿들을 때 reverse path가 있으면 rrep-ap를 1hop 내에 전송하여 2개의 Node-disjoint 주경로와 그들 사이의 여러 개의 백업 경로를 생성하여 링크 단절 시 복구에 사용하며, AODV-BRL(Backup Routing with LHF)[9]에서는 Hello 메시지에 목적지 노드의 정보를 추가하여 reverse route를 mesh구조로 생성하고, 링크 단절 시 upstream node는 BRRQ와 BRRP 전송을 통해 extended routing table에서 next node를 LHF방식으로 선택하여 복구한다. 또한 IBR-AODV (Implicit Backup Routing-AODV)[10]에서는 백업 노드들이 데이터 패킷을 엿듣다가, 일정시간(random back-off) 동안 ACK가 수신되지 않으면 RC(Route Change)를 전송하고, 이를 수신한 upstream node가 경로 테이블(Route table)을 갱신한 후 ACK를 전송하면 해당 백업 노드가 버퍼에 있는 데이터를 next node로 전송하도록 제안하였지만, 데이터 패킷의 재전송이 일어날 때마다 추가적인 제어패킷(RC, ACK)이 발생되고 random back-off time과 같은 delay가 발생되는 문제점이 있다.

AODV-TCBR(Time-Critical Backup Route)[11]에서는 시간에 따라 엿듣는 모든 RREP에 대한 정보를 대체 경로 테이블(Alternate route table)에 추가하고, 링크 단절 시 이 정보를 이용해 복구하며, AODV-nthBR[12]에서는 링크 단절 시 distance와 energy에 기반해 백업 노드들 중 next node를 선택하도록 제안하였다. 앞에서 언급한 AODV-BR의 문제점을 개선하려는 연구들은 대부분 네트워크 밀도가 높아지는 경우에 참여하는 백업 노드들이 많아지기 때문에 경로 복구 단계에서 next node를 선정하기 위한 기법을 제안하고 있다.

하지만 제안하는 대부분의 방법이 추가적인 제어 패킷을 사용하거나, 백업 노드의 성능을 평가해야하기 때문에 추가적인 라우팅 오버헤드와 전송 지연이 발생하는 것은 피할 수 없는 문제점을 지닌다[6-12].

Ⅲ. AODV-IB 라우팅 프로토콜

본 논문에서 제안하는 AODV-IB는 우선 AODV-I에서 제안한 인터럽트 메시지 방식을 기반으로 메시지별 처리절차를 추가하여 개선하고, 여기에 AODV-BR의 단점을 개선한 백업 경로 방식을 결합하여 경로 재탐색 빈도를 줄여 제어 패킷 오버헤드와 전송 지연을 개선함으로써 성능을 개선한다.

선행연구 AODV-I가 주기적인 Hello 메시지를 인터럽트 방식으로 대체하기 위해 상황에 따른 인터럽트 메시지를 정의하고, 이 메시지를 이용하여 이웃목록의 유지와 로컬 연결을 탐지하는데 그쳤다면, 제안하는 AODV-IB는 수신된 인터럽트 메시지를 이용하여 링크 단절이 예측되는 경우와 예측 불가능한 링크 단절이 탐지된 경우로 나누어 처리절차를 추가하여 성능을 개선하였다. 또한 AODV-IB에 결합한 개선된 백업 경로 기법은 AODV-BR 계열의 선행 연구들과 달리 백업 경로 정보를 저장하기 위해 추가적인 대체 경로 테이블을 생성하지 않고, 기존의 경로 테이블에 <BackupList>필드만을 추가하여 해당 경로의 백업 경로 정보를 저장한다. <BackupList>필드는 삽입과 삭제가 빠른 연결 리스트를 이용하여 미리 LHF(Least Hop count First)방식으로 저장하고, 경로 복구 시 백업 노드를 순차적으로 인출하여 이웃노드라면 next node로 선택하고 그렇지 않으면 다음 백업 노드를 인출하여 이웃노드 여부를 체크하여 경로 복구에 사용함으로써 next node를 선택하기 위한 추가적인 제어패킷과 지연 없이 경로 복구를 할 수 있도록 개선하였다.

인터럽트 메시지와 백업 경로 복구를 중심으로 작성된 AODV-IB의 처리도(Processing Diagram)는 그림 1과 같으며, 우선 각 노드는 시작과 동시에 시작 모듈(AodvInitiate)이 호출되면서 초기화 작업을 하고, 시작 이벤트가 발생한 것을 이웃 노드에 알리기 위해 <START> 메시지가 전송되도록 이벤트 처리 모듈(EventProcess)을 호출한다. 이후 잔존 에너지량을 체크하는 모듈(BetteryCheck)과 노드의 이동 상태를 체크하는 모듈(MobilityCheck)이 이벤트가 발생한 것을 감지할 때 이벤트 처리 모듈(EventProcess)을 호출하여 해당하는 인터럽트 메시지가 이웃 노드에 전송될 수 있도록 한다. 또한 전송된 인터럽트 메시지 및 제어 패킷을 수신한 각 노드는 메시지 처리 모듈(MessageProcess)을 통하여 각 메시지별 처리 절차에 따라 자신의 경로 테이블에 메시지를 전송한 이웃 노드에 대한 정보를 추가 또는 갱신한다.

마지막으로 노드가 전송할 데이터를 수신한 경우,

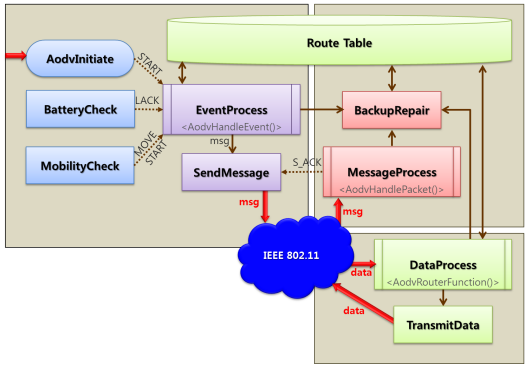

그림 1. 처리도
Fig. 1. Processing Diagram

각 노드는 데이터 처리 모듈(DataProcess)을 호출하여 경로가 없는 경우는 백업 경로 정보도 없기 때문에 즉시 경로 재탐색을 수행하고, 사용 가능한 경로가 있는 경우는 데이터를 전달하며, 경로는 있지만 해당 경로를 사용할 수 없는 경우는 백업 경로가 있다면 백업 복구 모듈(BackupRepair)을 통해 경로를 복구한 후 데이터를 전송하고, 백업 경로가 없다면 지역 복구 및 경로 재탐색 절차에 따라 처리한다.

전통적인 AODV는 경로 테이블에서 이웃 노드의 lifetime이 일정 주기로 설정되고, 이후 제어 패킷이 수신되지 않아 timeout이 발생하면, 해당 이웃 노드와 이웃 노드를 next hop으로 사용하는 경로의 active 값을 false로 설정한다. 이때 경로를 즉시 삭제하지 않고, 일정 시간이 지난 뒤에도 여전히 active 값이 false이면 경로 테이블에서 삭제된다. active 값이 false인 이웃 노드와 경로를 즉시 삭제하지 않는 이유는 [1]에서 언급한 경로 단절의 원인 중 여전히 전송범위 내에 있지만, 무선 환경의 특성으로 인해 일시적인 단절이 발생할 수 있기 때문이다. 일단 active 값을 false로 설정하고, 유예 시간 안에 해당 이웃 노드로부터 제어 패킷이 수신되면, 해당 경로의 정보가 갱신 될 때 active 값을 true로 설정함으로써 경로 탐색 절차를 통해 얻은 경로를 일시적인 단절로 인해 경솔하게 삭제하는 오류를 줄일 수 있게 된다.

따라서 경로의 active 값이 true이면 "사용 가능한 경로가 있는 경우"에 해당하며, 경로의 active 값이 false인 경우는 "경로가 있지만 해당 경로를 사용할 수 없는 경우"에 해당한다.

그림 1의 처리도에 관한 각 모듈별 상세 알고리즘은 부록(Appendix)에 추가하였고, 이후 이 장에서는 인터럽트 메시지를 정의한 선행연구 AODV-I에 관해 정리하고, AODV-IB에서 제안하는 개선된 백업 경로

방식에 대해 설명하며, 이를 기반으로 AODV-IB에 관해 기술한다.

3.1 인터럽트 메시지를 이용한 AODV-I

AODV-I(AODV using Interrupt message)는 전통적인 AODV에서 이웃 탐지 기법으로 사용하는 Hello 메시지를 인터럽트 메시지로 대체함으로써, 주기적인 Hello 메시지로 인한 문제점을 개선한 선행 연구이다[4].

[4]에서는 각 이동 노드가 자신의 배터리량을 모니터링 할 수 있고, 위치 이동 감지 센서를 통해 이동 사실을 감지해 낼 수 있다는 것을 전제로 하고 있다.

AODV-I에서 경로를 수집하는 RDP(Route Discovery Phase)의 동작과 제어 패킷은 AODV와 동일하고, 이웃 목록과 로컬 연결에 영향을 줄 수 있는 상황이 발생하면 전송하는 인터럽트 메시지들의 종류와 정의는 표 1과 같다.

표 1. 인터럽트 메시지의 분류
Table 1. Classification of Interrupt messages

State	Message type	Interval (Set timer)	neighbor's Lifetime
start (Join) network	START	X	MAX
receive "START"	S_ACK	X	MAX
low battery	LACK	X	0
movement	MOVE	O	ALLOWED_MESSAGE_LOSS * MESSAGE_INTERVAL

3.1.1 〈START〉와 〈S_ACK〉 메시지

<START>메시지는 두 가지 경우로 나누어 전송되는데 우선 그림 1의 시작 모듈(AodvInitiate)에서 전송하는 경우로 노드가 네트워크에 새로 진입(물리적인 고장 이후 재진입 포함)하는 이벤트의 발생을 알린다.

다음으로 노드의 이동 상태를 체크하는 모듈(MobilityCheck)에서 전송하는 경우로 해당 노드가 이동을 하다(주기적인 <MOVE> 메시지 전송) 멈추는 경우 멈춘 곳의 이웃 노드에게 해당 이벤트의 발생 사실과 함께 자신의 존재를 알리기 위해 전송된다.

이 두 경우에 전송하는 <START>메시지를 수신한 이웃 노드는 메시지 처리 모듈(MessageProcess)을 통해 <S_ACK>메시지로 응답하여 서로에 대한 정보를 주고받는다.

3.1.2 〈LACK〉 메시지

<LACK> 메시지는 그림 1의 잔존 에너지량을 체크하는 모듈(BatteryCheck)에서 배터리 용량이 부족한 경우 즉, 노드의 에너지 소모가 발생할 때마다 아직 <LACK> 메시지를 전송하지 않았다면, 노드의 잔존 에너지량이 Threshold 값 이하인지 체크함으로써 이벤트 발생 여부를 결정하고, 해당 이벤트가 발생한 경우 이벤트 처리 모듈(EventProcess)을 이용하여 이웃 노드에게 전송하는 메시지이다(부록<BatteryCheck>의 STEP 2~3 참조). 이를 수신한 이웃 노드들은 링크 단절을 예측할 수 있어 그림 1의 메시지 처리 모듈(MessageProcess)을 통해 자신의 경로 테이블에서 해당 노드의 lifetime을 0으로 설정하고, 일정시간 이후 이웃 목록(Neighbor set)에서 제거되도록 한다.

3.1.3 〈MOVE〉 메시지

<MOVE> 메시지는 위치 이동 감지 센서에 의해 노드의 위치 좌표 값이 변경되면, 이동이 발생한 것으로 판단하여, 그림 1에서 노드의 이동 상태를 체크하는 모듈(MobilityCheck)이 호출되고, 해당 모듈은 다음 <MOVE> 메시지의 전송을 위해 timer를 일정 주기로 설정함으로써, 해당 주기 마다 노드의 좌표 값을 체크하여, 변동 시 이벤트 처리 모듈(EventProcess)을 이용해 노드가 이동을 멈출 때(<START>메시지 전송)까지 주기적으로 전송하는 메시지이다(부록 <MobilityCheck>의 STEP 2~3 참조). 이를 수신한 이웃 노드들은 메시지 처리 모듈(MessageProcess)을 통해 자신의 경로 테이블에서 해당 노드의 lifetime과 timer를 일정 주기(ALLOWED_MESSAGE_LOSS * MESSAGE_INTERVAL)로 설정하고, <MOVE>메시지가 수신될 때 마다 갱신함으로써 이동 중인 해당 노드가 전송 범위에 있는지를 체크할 수 있다.

이 후 <START>메시지가 수신되면 해당 노드는 전송 범위 안에서 멈춘 것이고, 반면 timer가 만료될 때까지 다음 <MOVE>메시지나 <START>메시지가 수신되지 않으면 해당 노드는 전송 범위를 벗어난 것으로 경로 테이블에서 제거함으로써 이웃 목록에서도 제거된다.

Hello 메시지와 달리 AODV-I에서 정의된 인터럽트 메시지들은 그 종류를 통해 이웃 노드의 상태를 알 수 있기 때문에 링크 단절을 예측 또는 탐지하여 성능 개선을 꾀할 수 있으므로 각 인터럽트 메시지별 후속 처리에 관해 3. 인터럽트 메시지와 백업 경로기법을 이용한 AODV-IB에서 설명한다.

3.2 AODV-IB의 제안하는 백업 경로 기법

백업 경로 기법을 백업 경로 생성 단계와 링크 단절 시 경로 복구 단계로 나누어 보면 AODV-BR 계열의 선행 연구들은 대부분 경로 복구 단계에 중점을 두

고 있다. 즉, 백업 경로 생성 단계에서는 AODV-BR과 같이 RREP를 엿들어 수집한 백업 경로 정보를 대체 경로 테이블을 생성해 저장하고, 경로 단절 시 경로 복구 단계에서는 중복 데이터 전송 문제를 갖는 AODV-BR의 문제점을 보완하기 위해 백업 노드들 중 최적의 next node를 결정하는 다양한 방법들을 제안하고 있다[6-12]. AODV-IB에서 제안하는 개선된 백업 경로 방식은 선행 연구들과 달리 경로 복구 단계뿐만 아니라 백업 경로 생성 단계도 개선하였으며, 각 단계를 나누어 아래와 같이 설명한다.

3.2.1 백업 경로 생성 단계

AODV-IB의 백업 경로 생성 단계에서 백업 경로에 관한 정보는 AODV-BR와 같이 RREP를 엿들어 수집하지만, 선행연구들과 다른 점으로는 먼저 자료구조 측면에서 검색 시간을 줄이기 위해 대체 경로 테이블을 따로 운영하지 않고, 경로 테이블에 <BackupList>필드만 추가하여 운영한다는 것이다.

다음으로 <BackupList>필드는 그림 2와 같이 중간 지점에서 자료의 삽입과 삭제가 O(1)의 시간에 가능한 장점을 가진 연결리스트(Linked list)를 사용하여, 해당 경로에 대한 백업 노드 정보를 LHF(Least Hop count First)방식으로 저장함으로써, 경로 단절 시 추가적인 검색절차 없이 백업 노드를 순차적으로 인출하여 next node로 이용할 수 있어 복구된 경로의 길이와 지연을 최소화한다는 것이다.

LHF 방식의 사용은 선행 연구 AODV-BRL에서 Extended routing table에 저장된 백업 경로 정보들을 LHF 방식으로 검색해 최적의 대체노드(optimal substitute node)를 결정하면, 복구 지연(repair delay)을 최소화하고, 성공적인 고정율(successful fixing rate)을 향상시키며, 복구된 경로의 길이가 최소화되어 효율적인 복구를 제공한다는 연구 결과에 근거한다[9].

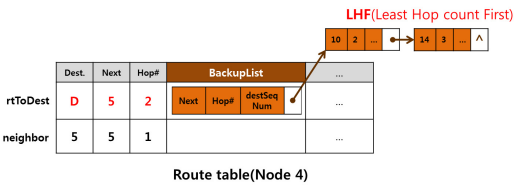

그림 2. AODV-IB의 Route table
Fig. 2. Route table of AODV-IB

3.2.2 백업 경로를 이용한 경로 복구 단계

AODV-IB의 백업 경로 생성 단계에서 백업 노드 정보를 연결리스트를 이용해 LHF 방식으로 저장했기 때문에 경로 복구 시 <BackupList>필드에서 별도의 검색 절차 없이 백업 노드를 순차적으로 인출하여 사용할 수 있다. 또한 인터럽트 메시지로 동적인 위상 변화를 반영한 이웃 목록과 로컬 연결 정보를 이용하여, 인출한 백업 노드가 이웃 목록에 속하는지 체크함으로써 경로 복구의 실패 가능성을 줄인다. 즉, 경로 복구 시 백업 노드가 여전히 이웃 노드인 경우 즉시 next node로 채택되어 경로 복구에 사용하고, 이웃 노드가 아니라면 <BackupList> 필드에서 다음 백업 노드를 인출하는 과정을 통해 빠르게 복구할 수 있기 때문에 next node를 선정하기 위한 추가적인 제어 패킷은 없고, 지연은 최소화하였으며 경로 복구의 성공 가능성도 높인다.

3.3 인터럽트 메시지와 백업 경로 기법을 이용한 AODV-IB

AODV-IB는 선행연구 AODV-I와 같이 인터럽트 메시지를 이용하여 이웃 목록 유지와 로컬 연결을 탐지하고, 데이터 전송 시 경로 탐색 단계(RDP)에서 주 경로의 정보를 수집할 때 앞서 기술한 AODV-IB에서 제안하는 개선된 백업 경로 기법을 이용하여 백업 경로에 관한 정보도 함께 얻는다. AODV-IB의 경로 유지 단계(RMP)에서 데이터 전송에 참여하는 각 노드들은 수신된 인터럽트 메시지를 이용하여 경로 상의 next node 상태를 알 수 있기 때문에 경로 단절이 예측되는 경우와 예측 불가능한 경우로 나누어 처리 절차를 추가하여 개선하였다. 즉, next node의 배터리 부족과 같이 경로 단절이 예측되고 발생 여부가 확실한 경우 백업 경로 정보를 이용하여 미리 경로를 수정함으로써 링크 단절이 발생하지 않도록 처리하거나, next node의 이동으로 인해 경로 단절이 예측은 되지만 발생여부가 불확실한 경우 백업 경로 정보를 이용해 데이터 패킷의 손실과 링크 단절을 미리 대비하도록 처리하여 링크 단절 발생 빈도를 줄일 뿐만 아니라 패킷 전달률도 향상시키도록 개선하였다. 또한 물리적인 고장, 무선 환경의 특성으로 인한 예측할 수 없는 경로 단절을 탐지한 경우 백업 경로 정보를 이용하여 경로를 빠르게 복구함으로써 경로 재탐색 빈도를 줄일 수 있도록 개선하였다. 따라서 AODV-IB에서 수신된 인터럽트 메시지별 처리 절차를 링크 단절이 예측되는 경우와 예측 불가능한 경우로 나누어 설명한다. 아울러, AODV-IB에서는 중복 데이터 패킷을 구별하기 위해 데이터 패킷의 헤더에 <seqNum>항목을 추가하고, 소스 노드가 데이터 패킷을 전송할 때 고유한 <seqNum> 값을 설정하여 전송함을 전제한다.

3.3.1 예측 가능한 링크 단절과 백업 경로를 이용한 경로 복구

(1) 배터리 부족으로 인한 링크 단절

데이터 전송 시 경로 상의 next node가 전송한 <LACK>메시지는 배터리 부족으로 인한 링크 단절을 예측할 수 있고, 그 발생 여부가 확실하기 때문에 이 메시지를 수신한 upstream node는 링크 단절이 발생하기 전에 백업 노드를 이용하여 경로를 우회 하도록 수정함으로써 링크 단절 발생 빈도를 낮출 수 있게 된다. 그림 1의 잔존 에너지량을 체크하는 모듈(BatteryCheck)과 이벤트 처리 모듈(EventProcess)을 통해 잔존 에너지량이 부족한 노드(노드4)는 그림 3의 (a)와 같이 <LACK>메시지를 브로드캐스트하고, 이를 수신한 이웃 노드들은 그림 1의 메시지 처리 모듈(MessageProcess)을 호출하여 각자의 경로 테이블에서 해당 노드(노드4)의 lifetime을 0으로 조정하여 일정시간 이후 이웃 목록에서 제거되도록 한다. 또한 그림 3의 (b)와 같이 데이터 전송 시 이웃 노드 중 upstream node(노드3)가 next node(노드4)로부터 이 메시지를 수신한 경우 해당 경로가 단절될 것을 미리 예측할 수 있어, 이 경우도 메시지 처리 모듈(MessageProcess)을 통해 next node를 백업 노드(노드8)로 교체함으로써 링크 단절이 발생되지 않도록 처리하여 링크 단절 발생 빈도를 줄이는 효과를 얻을 수 있다.

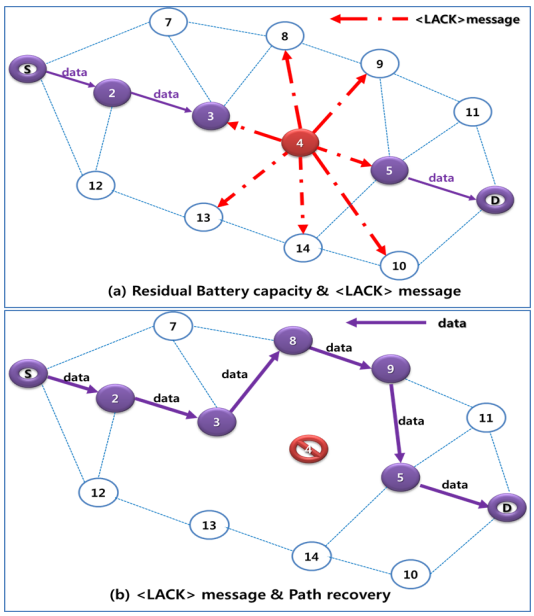

그림 3. 인터럽트(LACK) 메시지와 경로 복구
Fig. 3. Interrupt(LACK) message & Path recovery

(2) 노드의 이동으로 인한 링크 단절

next node의 이동으로 <MOVE>메시지를 수신한 경우, 링크 단절이 예상 되지만 그 발생 여부가 불확실한 상태이기 때문에 upstream node는 next node가 전송 범위 내에서 이동 중인 동안은 패킷 소실을 대비하여 백업 노드로도 중복 데이터를 함께 전송하고, 이때 중복 데이터를 수신한 노드들은 <seqNum>값을 참조하여 중복 데이터는 전달하지 않음으로써 중복 데이터로 인한 네트워크 혼잡을 제거한다. 이후 next node가 전송범위를 벗어나는 경우는 백업 노드로 경로를 빠르게 수정하여 계속 전송하고, 전송 범위 안에서 멈춘 경우는 다시 next node로만 데이터를 전송함으로써 링크 단절 빈도를 줄이는 것뿐만 아니라 노드의 이동으로 인한 패킷 소실을 줄여 패킷 전송률도 함께 향상시킬 수 있다.

이동 중인 노드(노드4)는 그림 1에서 이동 상태를 체크하는 모듈(MobilityCheck)과 이벤트 처리 모듈(EventProcess)을 통해 이동을 멈출 때(<START>메시지 전송)까지 그림 4의 (a)와 같이 <MOVE>메시지를 주기적으로 브로드캐스트하는데, 이를 수신한 이웃 노드들은 그림 1의 메시지 처리 모듈(MessageProcess)을 호출하여 각자의 경로 테이블에서 해당 노드의 lifetime과 timer를 일정주기로 설정하고, <MOVE>메시지가 수신될 때마다 갱신함으로써 이동 중인 해당 노드가 전송 범위 내에 있는지를 체크한다. 또한 데이터 전송 시 이웃 노드 중 upstream node(노드3)가 next node(노드4)로부터 이 메시지를 수신한 경우 next node의 이동으로 인한 링크 단절과 데이터 패킷의 소실 가능성이 있음을 예측할 수 있다. 따라서 next node(노드4)가 이동 중인 경우 이동을 멈추거나 (<START>메시지 전송) 전송 범위를 벗어날 때 (Timeout 발생)까지 그림 1의 데이터 처리 모듈(DataProcess)을 통해 그림 4의 (b)와 같이 백업 노드(노드8)로도 중복 데이터 패킷을 함께 전송하여 이동으로 인한 링크 단절과 데이터 패킷의 소실을 대비한다.

아울러 이후에 데이터 패킷을 수신한 노드들은 데이터 패킷 헤더의 <seqNum>을 참조하여 중복 데이터 패킷은 전달하지 않음으로 네트워크 혼잡으로 인한 피해를 최소화하면서 패킷 전송률을 향상 시킨다.

이 후 그림 5의 (C)와 같이 이동 중이던 next node (노드 4)로부터 <START>메시지가 수신되면 next node는 전송범위 안에서 멈춘 것이기 때문에 그림 1의 메시지 처리 모듈(MessageProcess)을 통해 경로

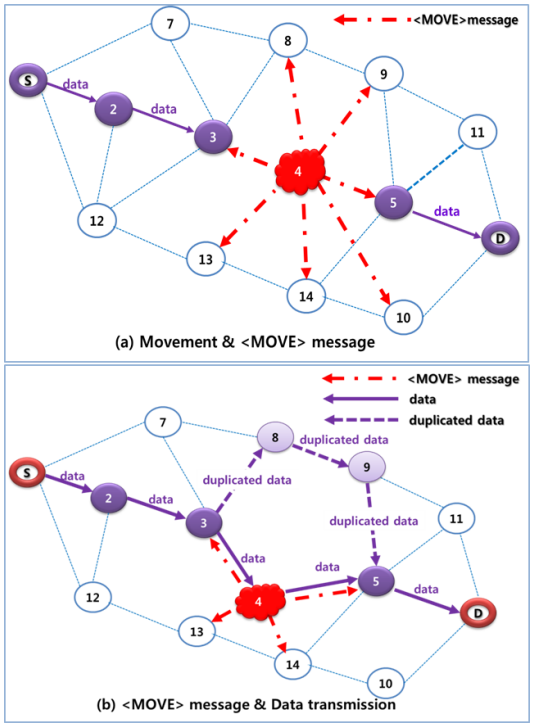

그림 4. 인터럽트(MOVE) 메시지와 데이터 전송
Fig. 4. Interrupt(MOVE) message & Data transmission

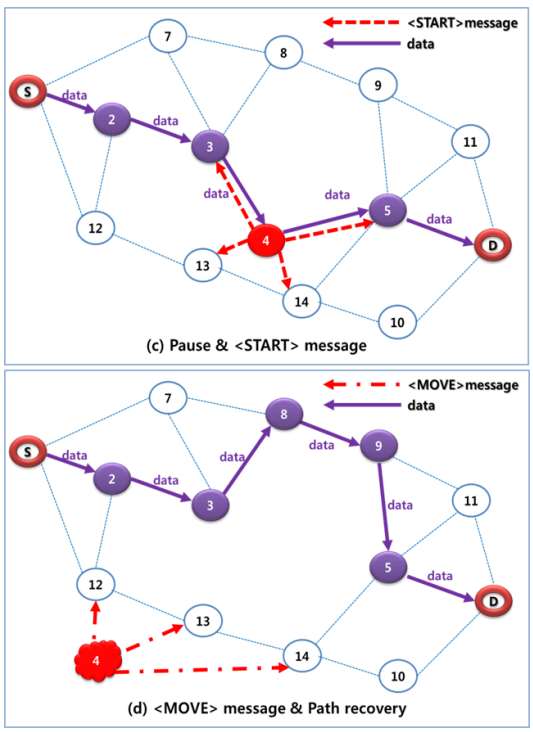

그림 5. 인터럽트(MOVE) 메시지와 경로 복구
Fig. 5. Interrupt(MOVE) message & Path recovery

테이블에서 next node의 정보를 갱신하고, 이동으로 인한 링크 단절과 데이터 소실의 가능성이 낮아져 upstream node(노드3)는 갱신된 경로 테이블의 정보와 그림 1의 데이터 처리 모듈(DataProcess)을 이용하여 이전과 같이 next node(노드4)로만 데이터를 전송한다.

반면, timer가 만료될 때까지 다음 <MOVE>메시지나 <START>메시지가 수신되지 않는다면, 그림 5의 (d)와 같이 next node는 전송 범위를 벗어난 것으로, Timer가 만료됨에 따라 그림 1의 이벤트 처리 모듈(EventProcess)이 호출됨으로써 upstream node(노드3)는 중복 데이터를 전송하던 백업 노드(노드8)를 next node로 빠르게 교체하여, 복구된 경로를 통해 데이터 처리 모듈(DataProcess)에 따라 데이터 전송을 계속함으로써 복구 지연을 줄인다.

3.3.2 예측 불가능한 링크 단절과 백업 경로를 이용한 경로 복구

데이터 전송 시 next node의 갑작스런 물리적 고장이나 무선 환경의 단절 등의 이유로 예측하지 못한 링크 단절이 발생한 경우 AODV-IB에서는 우선 upstream node가 물리적인 링크 단절을 탐지하고, 그림 1의 데이터 처리 모듈(DataProcess)을 통해 백업 노드를 이용한 경로 복구를 먼저 수행하여 경로 재탐색 빈도를 줄이도록 개선한다.

이 때 보유한 백업 노드를 모두 소진한 경우에도 경로 복구가 실패한다면, 다음으로 지역 복구(Local repair)의 수행 여부를 검토하고, 지역 복구도 불가한 경우에만 RERR을 전송하여 링크가 단절되었음을 주변에 알리고, 경로 재탐색(RDP)을 수행한다.

Ⅳ. 실험 분석

4.1 시뮬레이션 환경

인터럽트 메시지와 개선된 백업 경로 기법을 이용한 AODV-IB의 성능 분석을 위한 시뮬레이션 파라메터들은 [3]에서 제안하는 AODV 라우팅 프로토콜의 성능에 영향을 미치는 Factor들과 성능 메트릭들 간의 상관관계 분석을 기반으로 선정하였으며, 그 결과는 표 2와 같다.

4.2 성능 메트릭(Performance metrics)

본 연구에서 제안하는 AODV-IB의 성능을 분석하기 위해 성능 메트릭으로는 IETF MANET Working Group에 의해 제안된 패킷 전달률(PDR), 평균 종단

표 2. 시뮬레이션 파라메터
Table 2. Simulation parameter

Simulation parameter	Sim. 1	Sim. 2
Simulation Tool	QualNet 5.0	
Simulation Time	300 s	
Simulation Terrain	1500 m X 1500 m	
Number of Node	100 node	
CBR rate	**1 packet/sec**	**10 packet/sec**
CBR packet size	512 bytes	
Channel Frequency	2.4Ghz	
MAC layer	**IEEE802.11b**	**IEEE802.11g**
Bandwidth (Transmission range)	**2Mbps (449.403m)**	**24Mbps (252.717m)**
	11Mbps (272.450m)	
Mobility model	Random waypoint	
Maximum Velocity	**10(6～10)m/s, 20(16～20)m/s**	
Pause time	**30, 100 (s)**	
Number of mobile node	**10, 40 (%)**	
Message interval	**1, 3 (s)**	
Number of Low battery node	30 (%)	

간 지연, 제어 패킷 오버헤드가 사용되었으며, 그 외에도 네트워크 수명에 관련이 있는 잔존 에너지량과 특히 링크 단절 시 경로 복구가 실패하면 전송하는 RERR 패킷의 수 등을 사용하였다.

4.3 시뮬레이션 결과

시뮬레이션 결과는 선행연구인 인터럽트 메시지를 이용한 AODV-I와 본 논문에서 제안하는 AODV-IB를 성능 메트릭을 기준으로 비교 분석하였다. 또한 결과는 각 성능 메트릭 별로 시뮬레이션 1(IEEE 802.11b, CBR 1packet/sec)에서 대역폭이 2Mbps와 11Mbps의 결과와 시뮬레이션 2(IEEE 802.11g, CBR 10 packet/sec)에서 24Mbps의 결과인 세 가지 차트로 각각 제시된다. 각 차트의 X축은 우선 Pause time이 30, 100초로 구분되고, 이후 각 Pause time 내에서 이동 노드의 비율(#mobile)이 10, 40%로 분류되며, 각 이동 노드의 비율 내에서 <MOVE> message interval이 1, 3초로 구분되고, Message interval에서 Maximum velocity가 10, 20m/s로 분류되어 시뮬레이션 결과 값을 나타낸다.

4.3.1 패킷 전달률(PDR)

시뮬레이션 1에서 대역폭 2Mbps와 11Mbps의 PDR은 각각 그림 6, 그림 7과 같고, 시뮬레이션 2에서 대역폭 24Mbps의 PDR은 그림 8과 같다.

PDR 값은 본 연구에서 성능 개선의 주요 대상은 아니었음에도 백업 노드를 이용한 빠른 경로 복구를 통해 경로 단절 빈도를 감소시켜 패킷 소실을 줄임으로써 그림 6, 그림 7, 그리고 그림 8을 통해 AODV-I보다 AODV-IB의 PDR이 모두 높게 나타났다. 즉, AODV-I와 AODV-IB의 PDR 값을 자세히 분석해 보면, 우선 그림 6의 대역폭이 2Mbps인 경우 AODV-I에 비해 AODV-IB의 PDR 값이 평균 9% 향상되는 결과를 보였으며, 최소 5%에서 최대 17%의 상승률을 보였다. 또한 그림 7의 대역폭이 11Mbps인 경우 AODV-I에 비해 AODV-IB의 PDR 값이 평균 8% 향상 되었고, 상승폭은 최소 3%에서 최대 18% 까지로 나타났다. 그림 8의 대역폭이 24Mbps인 경우 AODV-I

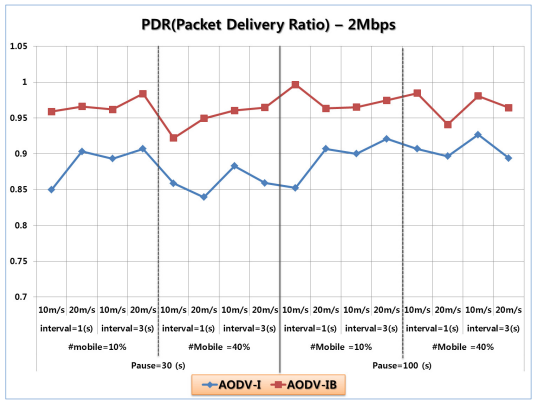

그림 6. 패킷 전달률 - IEEE 802.11b(2Mbps)
Fig. 6. Packet Delivery Ratio- IEEE 802.11b(2Mbps)

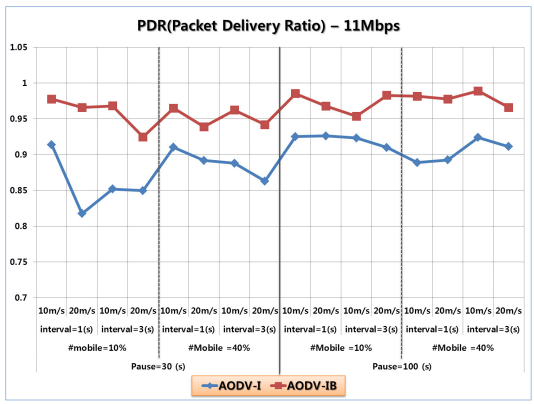

그림 7. 패킷 전달률 - IEEE 802.11b(11Mbps)
Fig. 7. Packet Delivery Ratio - IEEE 802.11b(11Mbps)

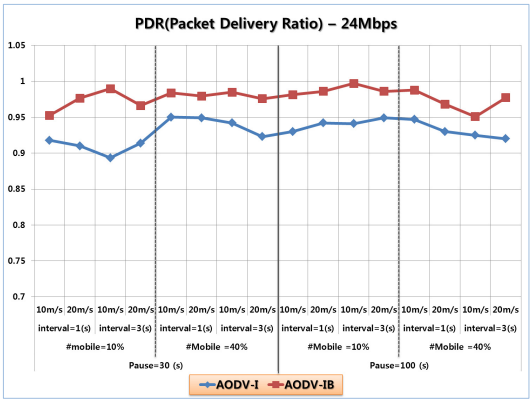

그림 8. 패킷 전달률 - IEEE 802.11g(24Mbps)
Fig. 8. Packet Delivery Ratio - IEEE 802.11g(24Mbps)

에 비해 AODV-IB의 PDR 값이 평균 5% 높았으며, 최소 3%에서 최대 11%의 차이를 보임으로써, 대역폭이 2Mbps, 11Mbps 그리고 24Mbps인 경우 모두 Pause time과 이동 노드의 비율, 메시지 주기, 최대 속도의 변화에 따라 각 구간에서, AODV-IB의 PDR값이 AODV-I의 값보다 높게 유지되는 것을 확인할 수 있다.

그림 7의 대역폭이 11Mbps인 경우는 2Mbps보다 대역폭은 증가하였으나 전송범위가 반감하면서, AODV-I는 그림 6(2Mbps)의 PDR 값 보다 11Mbps의 PDR 값이 일부 낮아지는 구간들이 발견되었지만, 대체로 AODV-I와 AODV-IB 모두 11Mbps의 PDR 값이 2Mbps의 값에 비해 증가되었다.

또한 대역폭 11Mbps의 전송범위를 비슷하게 유지하면서 대역폭을 24Mbps로 증가시키고, 트래픽의 양을 증가시킨 그림 8(24Mbps)의 PDR 값은 그림 7(11Mbps)의 값에 비해 AODV-I와 AODV-IB 모두 증가되는 것을 확인할 수 있다. 이것은 선행연구 AODV-I[4]에서 대역폭을 24Mbps로 증가시키고, 트래픽을 증가시킨 경우 전통적인 AODV의 PDR 값은 감소된 반면 주기적인 Hello 메시지를 인터럽트 메시지로 대체하여 불필요한 제어패킷을 최소화한 AODV-I의 PDR 값은 증가됨으로써 그 차이가 더 커진 실험 결과를 통해 본 연구의 대역폭 24Mbps에서 AODV-I와 이를 기반으로 한 AODV-IB 모두 트래픽의 증가에도 PDR 값이 증가된 근거를 확인할 수 있다.

4.3.2 평균 종단간 지연(Average end-to-end delay)

시뮬레이션 1에서 대역폭 2Mbps와 11Mbps의 평균 종단간 지연은 각각 그림 9, 그림 10과 같고, 시뮬레이션 2에서 대역폭 24Mbps의 평균 종단간 지연은 그림 11과 같다.

인터럽트 메시지와 백업 경로 기법을 이용해 경로 단절 빈도와 경로 재탐색 빈도를 최소화함으로써 전송 지연과 제어패킷 오버헤드를 개선하기 위한 AODV-IB의 평균 종단간 지연은 대역폭 2Mbps, 11Mbps 그리고 24Mbps 모두에서 선행연구 AODV-I의 평균 종단간 지연보다 낮게 유지되었다. AODV-I와 AODV-IB의 평균 종단간 지연을 자세히 분석해 보면, 우선 그림 9의 대역폭이 2Mbps인 경우 AODV-I에 비해 AODV-IB의 지연이 평균 46% 감소되는 결과를 보였으며, 최소 20%에서 최대 72%까지의 감소율을 보였다. 또한 그림 10의 대역폭이 11Mbps인 경우 AODV-I에 비해 AODV-IB의 지연이 평균 62% 감소되었고, 감소폭은 최소 30%에서 최대 86%까지로 나타났다. 그림 11의 대역폭이 24Mbps인 경우 AODV-IB의 지연이 평균 59% 감소되었으며, 최소 36%에서 최대 77%까지 차이를 보임으로써 그림 9~그림 11을 통해 AODV-I 보다 AODV-IB의 평균 종단간 지연이 Pause time과 이동 노드의 비율, 메시지 주기, 최대 속도의 변화에 따른 각 구간에서 모두 낮게 유지됨을 확인할 수 있다.

그림 10의 대역폭이 11Mbps일 때 평균 종단간 지연은 전송범위의 반감으로 인해 그림 9의 2Mbps일 때 보다 오히려 증가하였고, 이것은 AODV-I와 AODV-IB 모두에서 관찰되었다. PDR에서도 11Mbps인 경우 일부 구간에서 AODV-I의 PDR이 낮아지는 것이 관찰되었지만, 평균 종단간 지연에서는 전 구간에서 선행연구 AODV-I가 크게 증가되었고, 제안하는 AODV-IB의 평균 종단간 지연도 증가를 하였지만 백업 경로를 이용한 경로 복구로 AODV-I에

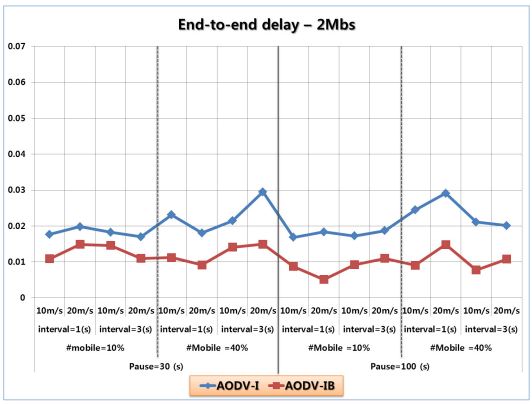

그림 9. 평균 종단간 지연-IEEE 802.11b(2Mbps)
Fig. 9. Average end to end delay-IEEE 802.11b(2Mbps)

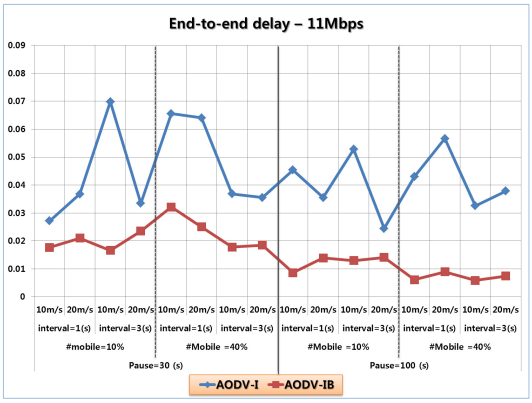

그림 10. 평균 종단간 지연-IEEE 802.11b(11Mbps)
Fig. 10. Average end-to-end delay-IEEE 802.11b(11Mbps)

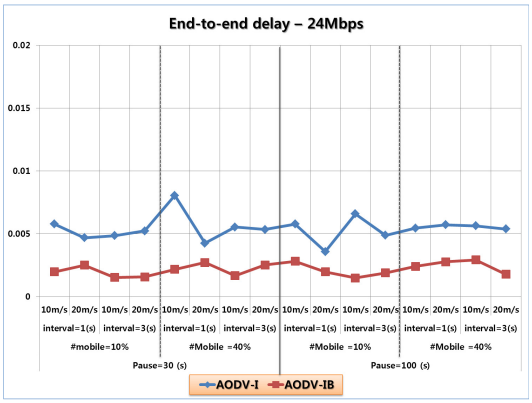

그림 11.평균 종단간 지연 - IEEE 802.11g(24Mbps)
Fig. 11. Average end to end delay-IEEE 802.11g(24Mbps)

비해 크게 증가되지는 않았다.

이것은 링크가 단절되면 패킷을 큐에 넣고 경로가 확보되면 그 때 다시 패킷을 꺼내 전송하기 때문에 경로 재탐색에 따른 지연시간이 발생하게 되고, 이처럼 단절된 링크가 바로 패킷 손실로 이어지지 않는 경우도 발생하기 때문에 PDR 값에 직접적인 영향을 미치기 보다는 오히려 평균 종단간 지연시간과 밀접한 관련이 있기 때문이다[4].

또한 11Mbps와 비슷한 전송범위로 대역폭과 트래픽의 양을 함께 증가시킨 그림 11(24Mbps)의 평균 종단간 지연에서는 트래픽의 증가에도 평균 종단간 지연이 AODV-I와 AODV-IB 모두 크게 감소하는 것을 확인할 수 있다.

4.3.3 제어 패킷 오버헤드(CPO : Control Packet Overhead)

시뮬레이션 1에서 대역폭 2Mbps와 11Mbps의 제어 패킷 오버헤드는 각각 그림 12, 그림 13과 같고, 시뮬레이션 2에서 대역폭 24Mbps의 제어 패킷 오버헤드는 그림 14와 같다.

제어 패킷 오버헤드에 영향을 주는 요인은 이동 노드의 비율, 메시지 주기, Pause time, 경로 단절로 인한 경로 재탐색 빈도 등이 있고, 제어 패킷의 종류로는 이동 노드의 비율과 메시지 주기에 따라 주기적으로 전송하는 MOVE 메시지, Pause time 마다 이동 노드가 멈추면 전송되는 START/S_ACK 메시지, 잔존 에너지량이 부족한 경우 전송되는 LACK 메시지 그리고 경로 단절 시 전송되는 RERR 패킷과 경로 탐색을 위해 전송되는 RREQ, RREP 패킷 등이 있다. AODV-I가 주기적인 Hello 메시지를 인터럽트 메시지로 대체하여 제어 패킷 오버헤드를 줄였다면, AODV-IB는 이 가운데 백업 경로 기법을 통해 경로 단절 빈도를 줄여 경로 재탐색 빈도를 최소화함으로써 RERR, RREQ 그리고 RREP 패킷을 줄여 성능을 개선한다.

AODV-IB의 제어 패킷 오버헤드는 대역폭 2Mbps, 11Mbps 그리고 24Mbps 모두에서 AODV-I에 비해 낮게 유지되었다. AODV-I와 AODV-IB의 제어 패킷 오버헤드에 관한 실험결과를 자세히 분석해 보면, 우선 그림 12의 대역폭이 2Mbps인 경우 AODV-I에 비해 AODV-IB의 CPO가 평균 11% 감소되는 결과를 보였으며, 감소폭은 최소 2%에서 최대 22% 까지로 나타났다. 또한 그림 13의 대역폭이 11Mbps인 경우 AODV-IB의 CPO가 평균 14% 감소되었고, 최소 3%에서 최대 47%까지의 감소율을 보였다. 마지막으로

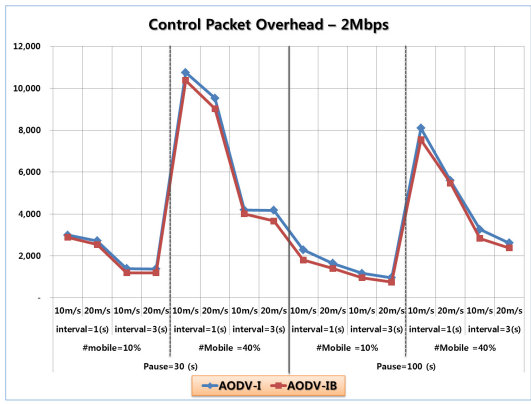

그림 12.제어 패킷 오버헤드 - IEEE 802.11b(2Mbps)
Fig. 12. Control Packet Overhead - IEEE 802.11b(2Mbps)

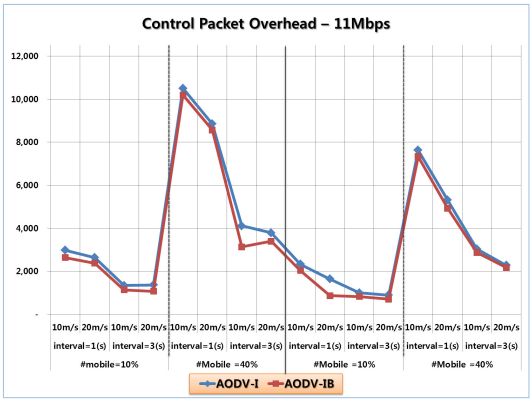

그림 13. 제어 패킷 오버헤드 - IEEE 802.11b(11Mbps)
Fig. 13. Control Packet Overhead - IEEE 802.11b(11Mbps)

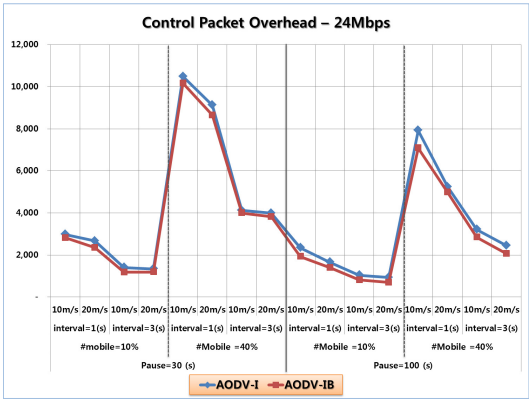

그림 14. 제어 패킷 오버헤드 - IEEE 802.11g(24Mbps)
Fig. 14. Control Packet Overhead - IEEE 802.11g(24Mbps)

그림 14의 대역폭이 24Mbps인 경우 AODV-IB의 CPO가 평균 11%감소되었으며, 최소 3%에서 최대 25%까지 차이를 보임으로써, 대역폭이 2Mbps, 11Mbps 그리고 24Mbps의 CPO인 그림 12~14를 통해 Pause time과 이동 노드의 비율, 메시지 주기, 최대 속도의 변화에 따라 모든 구간에서 AODV-IB의 제어 패킷 오버헤드가 선행연구 AODV-I 보다 낮게 유지되는 것을 확인할 수 있다.

제어패킷 오버헤드는 Pause time이 30초이고, 이동 노드의 비율이 40%이며, 메시지 주기가 1초인 경우 가장 높은 값을 나타내고 반면 Pause time이 100초이고, 이동 노드의 비율이 10%이며, 메시지 주기가 3초인 경우 가장 낮은 값을 나타내는데 이 두 가지 값의 차이가 너무 커서 차트 상의 단위가 커져 AODV-I와 AODV-IB가 겹쳐 보이지만 전 구간에서, 제안하는 AODV-IB의 제어패킷 오버헤드가 더 낮게 유지 되었다는 것을 다시 한 번 밝히며, 참고로 지면 관계상 본문에 싣지는 않았으나 링크 단절 시 전송하는 RERR 패킷의 수에서도 제안하는 AODV-IB가 2Mbps에서는 평균 60%, 11Mbps에서는 평균 63% 그리고 24Mbps에서는 평균 62% 더 낮게 발생되어, 경로 재탐색 빈도를 줄인 것을 확인하였음을 밝힌다.

또한 다른 성능 분석에서처럼 그림 13의 대역폭이 11Mbps인 제어패킷 오버헤드에서도 선행연구 AODV-I와 제안하는 AODV-IB의 제어패킷 오버헤드가 더 큰 차이를 보이는 구간들이 관찰되었다.

4.3.4 잔존 에너지량(Residual Battery Capacity)

시뮬레이션 1에서 대역폭 2Mbps와 11Mbps의 잔존 에너지량은 각각 그림 15, 그림 16과 같고, 시뮬레이션 2에서 대역폭 24Mbps의 잔존 에너지량은 그림 17과 같다.

잔존 에너지량에 영향을 주는 주요 요인으로는 대역폭, PDR, 제어 패킷 오버헤드, 트래픽 등이 있다. AODV-I와 비교하여 AODV-IB는 경로 단절을 예측 또는 탐지 시 백업 경로를 이용하여 빠른 경로 복구를 수행함으로써 경로 단절 빈도를 낮추게 된다. 따라서 경로 단절 빈도가 감소되면 경로 단절 시 발생하는 패킷 소실도 줄어 패킷의 재전송을 위한 에너지 소모를 줄일 수 있고, 경로를 재탐색하는 횟수를 줄일 수 있어, 경로 단절 및 탐색을 위해 발생되는 RERR, RREQ 그리고 RREP 패킷의 전송을 위한 에너지도 절약할 수 있기 때문에 AODV-IB의 잔존 에너지량은 더 좋아지게 되며, 이 사실은 그림 15, 그림 16 그리고 그림 17을 통해 AODV-I 보다 AODV-IB의 잔존 에너지량이 모두 높게 나타나는 것으로 확인할 수 있다.

즉, AODV-I와 AODV-IB의 잔존 에너지량을 자세히 살펴보면, 우선 그림 15의 대역폭이 2Mbps인 경우 AODV-I에 비해 AODV-IB의 잔존 에너지량이 평균

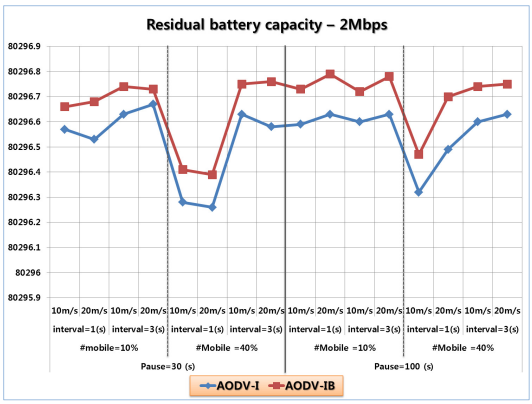

그림 15. 잔존 에너지량 - IEEE 802.11b(2Mbps)
Fig. 15. Residual Battery Capacity - IEEE 802.11b(2Mbps)

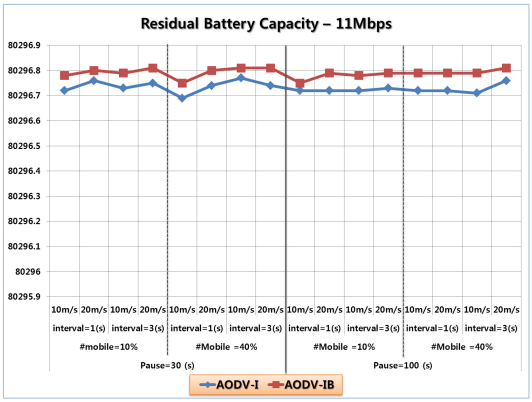

그림 16. 잔존 에너지량 - IEEE 802.11b(11Mbps)
Fig. 16. Residual Battery Capacity-IEEE 802.11b(11Mbps)

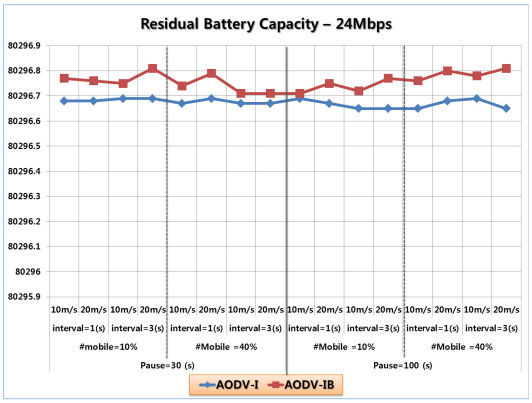

그림 17.잔존 에너지량 - IEEE 802.11g(24Mbps)
Fig. 17. Residual Battery Capacity-IEEE 802.11g(24Mbps)

0.135(mAhr) 향상되는 결과를 보였으며, 최소 0.06(mAhr)에서 최대 0.21(mAhr)의 상승률을 보였다. 또한 그림 16의 대역폭이 11Mbps인 경우 AODV-IB의 잔존 에너지량이 평균 0.059(mAhr) 향상 되었고, 상승폭은 최소 0.03(mAhr)에서 최대 0.08(mAhr)까지로 나타났다.

마지막으로 그림 17의 대역폭이 24Mbps인 경우 AODV-IB의 잔존 에너지량이 평균 0.086(mAhr) 높았으며, 최소 0.02(mAhr)에서 최대 0.16(mAhr)의 차이를 보임으로써, 대역폭이 2Mbps, 11Mbps 그리고 24Mbps인 경우 모두 시뮬레이션 파라메터들의 변화에 따른 각 구간에서 AODV-IB의 잔존 에너지량이 AODV-I 보다 높게 유지되는 것을 확인하였으며, 아울러 잔존 에너지량은 네트워크 수명과도 관련이 있어 네트워크 수명이 증가됨을 함께 확인할 수 있다.

분석의 결과 본 연구의 의의는 선행연구 AODV-I에서 제안한 인터럽트 메시지 방식을 기반으로 이웃 목록과 로컬 연결을 탐지하는데 머물지 않고, 각 인터럽트 메시지의 종류에 따라 링크 단절의 예측 및 탐지에 따른 처리 절차를 추가하여 개선한 AODV-IB를 제시한 점이다. 즉, 물리적 고장, 무선 환경 특성 등으로 인한 예측 불가능한 실패와 노드의 이동, 부족한 배터리로 인한 예측 가능한 실패의 링크 단절을 예측 또는 탐지하여, 개선된 백업 경로 기법을 이용한 빠른 경로 복구로 패킷 전달률을 높이고, 경로 단절 및 경로 재탐색의 빈도를 줄임으로써 전송 지연과 제어 패킷 오버헤드를 최소화하여 성능을 개선하였다는 것이다.

또한 AODV-BR과 이를 개선하려는 선행연구들과 달리 AODV-IB에서는 별도의 대체 경로 테이블을 생성하지 않고 기존 경로 테이블에 <BackupList>필드만 추가하여, 수집된 백업 경로 정보를 연결 리스트를 이용해 LHF방식으로 저장하고, 경로 단절 시 추가적인 제어패킷 없이 순차적으로 인출한 백업 노드로 빠르게 경로 복구를 수행할 수 있도록 개선하는 결과를 가져왔다.

V. 결 론

MANET 환경에서 라우팅 프로토콜은 노드의 이동성으로 인한 동적인 위상 변화와 배터리 상태 그리고 무선 환경의 특성 등으로 인한 빈번한 링크 단절에 대해 민첩하고 적절하게 대응해야 한다. 하지만 전통적인 AODV 라우팅 프로토콜은 단일 경로와 주기적인 Hello 메시지로 인해 빈번한 링크 단절에 대해 효율적으로 대응할 수 없어 개선이 필요하다. 주기적인 Hello 메시지를 인터럽트 방식으로 대체하여 성능을 개선한 선행연구 AODV-I에서 정의한 인터럽트 메시지들은 Hello 메시지와 달리 그 종류에 따라 이웃 노드의 상태를 알 수 있어, 링크 단절을 탐지할 뿐만 아니라 예측도 할 수 있기 때문에 각 메시지 종류에 따라 처리 절차를 추가하여 성능을 더욱 개선할 수 있다. 또한 AODV-I는 단일 경로를 사용함으로써 경로 재탐색 빈도를 높이는 문제점을 여전히 가지고 있어 링크 단절로부터 효율적인 경로 복구를 제공하는 백업 경로 기법을 추가한 성능 개선도 가능하다.

이에 본 연구에서 제안하는 AODV-IB에서는 AODV-I의 인터럽트 메시지방식을 기반으로 동적인 위상 변화를 반영한 이웃 목록과 로컬 연결 정보를 유지하고, 데이터 전송 시 수신된 인터럽트 메시지의 종류에 따라 링크 단절이 예측되거나 탐지된 경우 개선된 백업 경로방식을 접목하여 추가적인 제어패킷 없이 빠르게 경로 복구를 수행함으로써 경로 재탐색 빈

도와 전송 지연을 최소화하여 성능을 개선하였다.

AODV-IB의 성능 분석은 QualNet 5.0을 이용하였고, 선행 연구에서 제안하는 시뮬레이션 파라메터를 선정하여, 이를 기반으로 수행한 실험 결과를 4가지 성능 메트릭을 기준으로 분석하였다. 분석결과 AODV-IB의 패킷 전달률이 각 파라메터들의 변화에 따라 높은 값을 유지하였고, 평균 종단간 지연에서도 AODV-I에 비해 낮은 값을 가졌다. 또한 AODV-IB의 제어 패킷 오버헤드에서도 AODV-I와 비교해 낮은 값을 유지 하였으며, 이로 인해 높은 잔존 에너지량을 보임으로써 4가지 성능 메트릭을 통한 분석 결과를 기반으로 AODV-IB가 AODV-I보다 전반적으로 우수함을 보였다.

본 연구는 선행연구 AODV-I에서 제안한 인터럽트 메시지 방식을 기반으로 이웃 목록과 로컬 연결을 탐지하고, 수신된 인터럽트 메시지의 종류에 따라 처리 절차를 추가하여 개선함으로써 링크 단절을 예측하거나 탐지한 경우 개선된 백업 경로 기법을 이용한 빠른 경로 복구로 경로 재탐색 빈도를 줄여, 전송 지연과 제어 패킷 오버헤드를 최소화하였다. 또한 AODV-BR을 기반으로 하는 선행연구들과 달리 AODV-IB에서는 기존의 경로 테이블에 백업 경로 정보를 연결 리스트를 이용해 LHF방식으로 저장하고, 경로 단절 시 순차적으로 인출한 백업 노드를 이용해 추가적인 제어패킷 없이 경로 복구를 수행할 수 있도록 개선하였다.

향후 인터럽트 메시지방식과 백업 경로기법을 결합한 환경에서 AODV의 다른 문제점들도 해결가능한지를 연구할 수 있는 기반을 마련함과 동시에 주기적인 beacon을 사용하는 요구기반 방식의 다른 라우팅 프로토콜에도 확장 적용이 가능할 것이다.

References

[1] J. H. Seo, K. H. Kim, and H. G. Seo, "An efficient route maintenance scheme utilizing hello messages for AODV - based ad hoc networks," *J. KISS,* vol. 31, no. 3, pp. 280-288, Jun. 2004.

[2] H. G. Seo, K. H. Kim, and J. H. Seo, "AFLRS : An AODV - based fast local repair scheme in ad hoc networks," *J. KISS,* vol. 31, no. 1, pp. 81-90, Feb. 2004.

[3] Y. Lee and J. Kim, "Performance comparison between routing protocols based on the correlation analysis of performance metrics for AODV routing protocol," *J. KITS*, vol. 12, no. 4, pp. 349-367, Dec. 2013.

[4] Y. Lee and J. Kim, "Performance enhancement of AODV routing protocol Using interrupt message in MANET," *J. KICS*, vol. 38, no. 10, pp. 785-800, Oct. 2013.

[5] S.-J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," *Wirel. Commun. and Netw. Conf.(WCNC)*, vol. 3, pp. 1311-1316, Chicago, USA, Sept. 2000.

[6] W. K. Lai, S.-Y. Hsiao, and Y.-C. Lin, "Adaptive backup routing for ad-hoc networks," *Comput. Commun.*, vol. 30, no. 2, pp. 453-464, Jan. 2007.

[7] J. Liu and F. Li, "An improvement of AODV protocol based on reliable delivery in mobile ad hoc networks," in *Proc. 5th Int. Conf. Inf. Assurance and Security (ISA '09)*, vol. 1, pp. 507-510, Xian, China, Aug. 2009.

[8] S. Wang, Q. Li, Y. Jiang, and H. Xiong, "Stable on-demand multipath routing for mobile ad hoc networks," *Asia-Pacific Conf. Computational Intell. and Ind. Appl.(PACIIA 2009)*, vol. 2, pp. 318-321, Wuhan, China, Nov. 2009.

[9] L. Yujun and H. Lincheng, "The research on an AODV-BRL to increase reliability and reduce routing overhead in MANET," *Int. Conf. Comput. Appl. and Syst. Modeling (ICCASM)*, vol. 12, pp. 526-530, Taiyuan, China, Oct. 2010.

[10] J. Jeon, K. Lee, and C. Kim, "Fast route recovery scheme for mobile ad hoc networks," *Int. Conf. Inf. Netw. (ICOIN)*, pp. 419- 423, Barcelona, Spain, Jan. 2011.

[11] M. H. Cheng, R. J. Deng, W. S. Hwang, and Y. J. Wu, "An AODV-based backup routing mechanism for time-critical in wireless mesh networks," *10th Int. Conf. Intell. Inf. Hiding and Multimedia Signal Process. (IIH-MSP)*, pp. 546-549, Kitakyushu, Japan, Aug. 2014.

[12] M. Rao and N. Singh, "Quality of service enhancement in MANETs with an efficient routing algorithm," *IEEE Int. Advance*

Comput. Conf. (IACC), pp. 381-384, Gurgaon, India, Feb. 2014.

[13] Y.-C. Lin and C.-W. Ke, "Adaptive route selection in mobile ad hoc networks," *4th Int. Conf. Commun. and Netw. in China, 2009. (ChinaCOM 2009),* pp. 1-5, Xian, China, Aug. 2009.

[14] T.-C. Huang, S.-Y. Huang, and L. Tang, "AODV-based backup routing scheme in mobile ad hoc networks," *Int. Conf. Commun. and Mob. Comput. (CMC)*, vol. 3, pp. 254-258, Shenzhen, China, Apr. 2010.

[15] S. Ding and L. Liu, "A node-disjoint multipath routing protocol based on AODV," *9th Int. Symp. Distrib. Comput. and Appl. to Business Eng. and Sci. (DCABES),* pp. 312-316, Hong Kong, China, Aug. 2010.

[16] T. S. Kumaran and V. Sankaranarayanan, "Early detection congestion and control routing in MANET," *7th Int. Conf. Wirel. and Optical Commun. Netw. (WOCN)*, pp. 1-5, Colombo, Sri Lanka, Sept. 2010.

[17] C. J. Hwang, A. Kush, and S. Taneja, "Making MANET energy efficient," *Global Mob. Congress (GMC)*, pp. 1-6, Shanghai, China, Oct. 2011.

[18] A. T. Thai and M. K. Kim, "A routing protocol with fast-recovery of failures using backup paths on MANETs," *J. KICS*, vol. 16, no. 7, pp. 1541-1548, Jul. 2012.

[19] Y. Wang and J. Liu, "A backup multipath routing protocol for ad hoc networks with dynamic topology," *3rd Int. Conf. Syst. Sci. Eng. Design and Manufacturing Inf. (ICSEM)*, pp. 237-241, Chengdu, China, Oct. 2012.

[20] G. Rajkumar and K. Duraiswamy, "Streamlined short alternate path protocol for Mobile Ad hoc Networks," *Int. Conf. Emerging Trends in Comput., Commun. and Nanotechnol. (ICE- CCN)*, pp. 452-455, Tirunelveli, India, Mar. 2013.

[21] C. Nishanthini, G. Rajkumar, and G.N. Jayabhavani, "Enhanced performance of AODV with power boosted alternate path," *Int. Conf. Comput. Commun. and Informatics (ICCCI)*, pp. 1-4, Coimbatore, India, Jan. 2013.

Appendix

< AodvInitiate >

Step 1 : Start event is occurred!

```
/* To avoid retransmission of START message */
set node->sendStart to FALSE;

/* To process StartEvent */
set event to INT;
set type to START;
call AodvHandleEvent(event, type);
set node->sendStart to TRUE;
```

< BatteryCheck >

Step 1 : To initialize battery parameters

```
/* To avoid retransmission of LACK message */
set node->battery->sendLack to FALSE;

/* To check remaining charge of battery */
set node->battery->lack to FALSE;

/* check Battery event */
if( (!node->battery->sendLack ) &&
   (node->battery->remaining  < Threshold) ) then
       call BatteryProcessEvent();
```

Step 2 : To decrease battery charge for the load

```
if(node->battery->remaining  > 0) then
{   node->battery->remaining  -= cost * duration;
    /* To check Battery event */
    if( (!node->battery->sendLack ) &&
       (node->battery->remaining  < Threshold) ) then
        call BatteryProcessEvent();
}
```

Step 3 : To process Lack events of Battery

```
Func BatteryProcessEvent( )
{    /* Lack event is occurred! */
     set node->battery->lack to TRUE;
     set event to INT;
     set type to LACK;
     /* To send LACK message  */
     call AodvHandleEvent(event, type);
     set node->battery->sendLack to TRUE;
}
```

< EventProcess >

Step 1 : To process for each Event type

```
Func AodvHandleEvent(event, type)
{
   switch(event)
      case CheckNeighbor_Timeout :
        /* Check connectivity based on Interrupt message */
          if(rtToDest->backuplist.size  >0)
              call BackupRepair(rtToDest);
          else
              send RERR packet for link failure

      case CheckRoute_Timeout :
        /* Remove the route that has not been used for awhile */
          if(rtToDest->backuplist.size  >0)
              call BackupRepair(rtToDest);
          else
              disable the route
                  and then delete it after delete period

      case INT :
      {   /* To broadcast Interrupt message */
          swicth(type)
              case START :
                 call SendMessage (msg);
              case LACK :
                 call SendMessage (msg);
              case MOVE :
                  call SendMessage (msg);
      }
}
```

Step 2 : To broadcast message

```
Func SendMessage (msg)
{   for(i=0; i< node->numberInterfaces ; i++)
        send Raw message to Mac
}
```

< MobilityCheck >

Step 1 : To initialize mobility parameters

```
/* To check mobility */
set node->mobility->moving to FALSE;
/* To avoid retransmission of START message */
set node->mobility->sendStart to FALSE;
```

Step 2 : Mobility event is occurred!

```
set node->mobility->moving to TRUE;
set node->mobility->sendStart to FALSE;
/* To set Timer for Move message
if( isExpireTimerSet == FALSE ) then
{   set isExpireTimerSet to TRUE;
    set interval to MESSAGE_INTERVAL;
    set event to MOVE;
    AodvSetTimer(interval, event);
    call MobilityProcessEvent( );
 }
```

Step 3 : To process Mobility events

```
Func MobilityProcessEvent( )
{   if(node->mobility->moving) then
    {   /* To send MOVE message */
        set event to INT;
        set type to MOVE;
        call AodvHandleEvent(event, type);
        set node->mobility->moving to FALSE;
     }
    else if ( !node->mobility->sendStart ) then
    {   /* To send START message */
        set event to INT;
        set type to START;
        call AodvHandleEvent(event, type);
        set node->mobility->sendStart to TRUE;
       set isExpireTimerSet to FALSE ;
    }
}
```

Step 4 : To process (Mobility) events by periods

```
Func AodvSetTimer(interval, event)
{
    if(isExpireTimerSet) then
    {   switch(event)
            case MOVE :
                call MobilityProcessEvent( );
            case CheckNeighbor_Timeout :
                call AodvHandleEvent(event, type);
            case CheckRoute_Timeout :
                call AodvHandleEvent(event, type);
     }
}
```

< DataProcess >

Step 1 : Determine the routing action to take for the given data packet

```
Func AodvRouterFunction(data)
{ /* source node of the route */
    if (isSource) then
    { /* Source node has a valid route to the destination */
        if(isValidRt) then
        { /* transmits a data to the next hop */
          TransmitData(data, rtToDest->nextHop);
          if(rtToDest->backuplist.size>0 && rtToDest->INT =="M" ) then
            /* transmit the duplicate data to the backup node */
              TransmitData(data, rtToDest->backuplist);
        }
        else
          /* There is no route to the destination and RREQ has not been sent */
            Initiate a RREQ packet
    }
 /* Intermediate node or destination of the route */
    else
      AodvHandleData(data, rtToDest);
}
```

Step 2 : Processing procedure when data is received from another node

```
Func AodvHandleData(data, rtToDest)
{   if (rtToDest) then
    { /* There is a valid route towards the destination */
        if(isValidRt) then
        { /* transmits a data to the next hop */
            TransmitData(data, rtToDest->nextHop);
            if(rtToDest->backuplist.size>0 && rtToDest->INT =="M" ) then
              /* transmits the duplicate data to the backup node */
                TransmitData(data, rtToDest->backuplist);
        }
      /* There is no valid route */
        else if(rtToDest->backuplist.size >0) then
                BackupRepair(rtToDest);
            else if (isLocalRepair) then
                    initiate RREQ packet for local repair
                else
                    send RERR packet for link failure
    }
  /* There is no route towards the destination of the route */
    else
        send RERR packet for unreachable destination
}
```

Step 3 : Forward the data packet to the next hop

```
Func TransmitData(data, nextHop)
{
        send data Packet to Mac layer
}
```

< MessageProcess >

Step 1 : called when AODV packet is received from MAC

```
Func AodvHandlePacket(msg)
{   switch(msg->event)
        case RREQ :
            handle request
        case RREP :
            handle reply
        case RERR :
            handle error
        case INT :
            call AodvHandleInterrupt(msg)
}
```

Step 2 : Processing procedure when Interrupt message is received

```
Func AodvHandleInterrupt(msg)
{   /* srcNode that sent this Interrupt message */
    switch(msg->type)
    {   case START :
            /* To modify lifeTime & INT of srcNode in Route table */
            set rtToEntry->lifetime to MAX_TIME;
            set rtToEntry->INT to "";
            / * To respond to START message */
            set msg->event to INT;
            set msg->type to S_ACK;
            call SendMessage (msg);
        case LACK :
            /* To remove srcNode entry in Route table */
            set rtToEntry->lifetime to 0;
            set rtToEntry->INT to "L";
           /* To repair paths(rtToDest) that use srcNode as a next hop*/
            for(i=0; i<RouteTableSize ; i++)
              if(rtToDest[i]->backuplist.size >0 &&
                rtToDest[i]->nextHop == srcNode ) then
                  BackupRepair(rtToDest[i]);
        case MOVE :
            set rtToEntry->lifetime to ALLOWED_MESSAGE_LOSS
                                    * MESSAGE_INTERVAL;
            set rtToEntry->INT to "M";
            /* To set Timer for Move message
            if( isExpireTimerSet == FALSE ) then
            {   set isExpireTimerSet to TRUE;
                set interval to rtToEntry->lifetime ;
                set event to CheckNeighbor_Timeout;
                AodvSetTimer(interval, event);
            }
        case S_ACK :
            set rtToDest->lifetime to MAX_TIME;
            set rtToDest->INT to "";
    }
    insert or update an rtToEntry(srcNode) into the Route table
}
```

Step 3 : To repair paths by backup node

```
Func BackupRepair(rtToDest)
{   for(i=0; i<rtToDest->backuplist.size ; i++)
        /* get next hop from <BackupList> field and check neighbor set */
        if ( isNeighbor) then  break;
}
```

이 윤 경 (Yun-kyung Lee)

2000년 2월 : 숙명여자대학교 전자계산학과 졸업
2008년 2월 : 숙명여자대학교 전자계산교육과 교육학석사
2008년 9월~현재 : 숙명여자대학교 컴퓨터과학과 박사과정
<관심분야> 운영체제, MANET, Routing protocol, IOT

김 주 균 (Ju-gyun Kim)

1985년 : 서울대학교 계산통계학과졸업
1985년~1986년 : DEC Korea 근무
1988년 : 서울대학교 계산통계학과 계산학석사
1992년 : 서울대학교 계산통계학과 계산학박사
1992년~현재 : 숙명여자대학교 이과대학 컴퓨터과학전공 교수
<관심분야> 운영체제, 성능평가, Caching Strategies, Flash Memory Technologies, MANET