DOI: 10.3745/KIPSTC.2008.15-C.6.531

애드 혹 네트워크에서 인접 행렬 기반의 라우팅 연구

이 성 수[†] · 김 정 미[††] · 박 희 주[†††] · 김 종 근[††††]

요 약

애드 혹 네트워크의 동적인 환경에서는 네트워크 토폴로지의 변화로 잦은 경로 단절과 이로 인한 전송 지연이 일어난다. 따라서 전송 지연에 민감한 애드 혹 네트워크의 실시간 환경에서는 효율적인 라우팅 방법이 주요 관심이 될 수밖에 없다. 그래프를 이용하는 통신 관련 이론의 주요 관심 중 하나는 주어진 노드들 중에서 어떠한 노드들이 연결되어 있으며, 최소 비용을 가진 경로는 어떻게 쉽게 찾을 것인가 하는 것 들이다. 애드 혹 네트워크에서 노드간의 연결은 인접 행렬로 나타낼 수 있다.

본 논문에서는 이러한 인접 행렬에 기반한 일련의 행렬 연산을 이용한 경로 검색 기법을 제안한다. 인접 행렬의 연산을 통해 구해진 연결 행렬을 이용하여 목적지로부터 소스까지 경로를 구하는 방법이다. 최단 경로를 검색하기 위한 인접 행렬 기반의 역검색 방법과 노드-비중첩 다중 경로를 검색하기 위한 인접 행렬 기반의 노드-비중첩 다중 경로 역검색 방법을 제안한다.

키워드 : 애드 혹 네트워크, 리더 선출, 인접 행렬, 경로 검색

A Study of Routing based on Adjacency Matrix in Ad hoc Networks

Sungsoo Lee[†] · Jeongmi Kim[††] · Heejoo Park[†††] · Chonggun Kim[††††]

ABSTRACT

With the dynamic and mobile nature of ad hoc networks, links may fail due to topology changes. So, a major challenge in ad hoc network is dynamically to search paths from a source to destination with an efficient routing method, which is an important issue for delay-sensitive real-time application. The main concerns of graph theory in communications are finding connectivity and searching paths using given nodes. A topology of the nodes in ad hoc networks can be modeled as an adjacency matrix.

In this paper, based on this adjacency matrix, we propose new path search algorithms using a sequence of matrix calculation. The proposed algorithms can search paths from a destination to a source using connectivity matrix. Two matrix-based algorithms for two different purposes are proposed. Matrix-Based Backward Path Search(MBBS) algorithm is designed for shortest path discovery and Matrix-Based Backward Multipath Search(MBBMS) algorithm is for multipath search.

Keywords : Ad Hoc Networks, Leader Election, Adjacency Matrix, Path Search

1. 서 론

다중 홉 무선 링크로 구성된 애드 혹 네트워크는 네트워크 토폴로지의 잦은 변화로 인한 경로 단절과 이로 인한 전송 지연이 불가피하다. 애드 혹 네트워크에서 제공되는 서비스는 FTP나 email과 같은 delay-tolerant 서비스뿐만 아니라 delay-sensitive한 실시간 어플리케이션까지도 포함될 수 있다. 그러므로 애드 혹 네트워크에서 이러한 delay-sensitive 어플리케이션을 위해서는 소스로부터 목적지까지의 효율적인 경로 설정이 중요하다.

최근 통신의 연구에 있어서 그래프 이론을 이용한 연구가 활발히 연구되고 있다. 애드 혹 네트워크에서 그래프 이론을 이용하는 경우의 주요 관심은 네트워크에서 노드들이 어떻게 연결되어 있으며, 두 노드간의 최소 비용을 가진 경로를 어떻게 설정하는가 하는 것들이다. 따라서 소스 노드로부터 시작되는 연결성 결정 그래프에 관한 알고리즘은 애드 혹 네트워크에서 주요한 이슈가 되고 있다. 애드 혹 네트워크에서 노드간의 연결 상태는 인접 행렬로 표현할 수 있다. 애드 혹 네트워크에서 인접 행렬을 이용하여 네트워크의 토폴로지를 나타내고 이를 네트워크 성능 평가에 이용하는 몇 가지의 방법들이 제안되었다. J. Zhang과 W. Seah는 인접 행렬을 이용하여 애드 혹 네트워크의 연결성 성능을 계산하

※ 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신 인력양성사업으로 수행된 연구 결과임
† 준 회 원 : 영남대학교 컴퓨터공학과 박사
†† 준 회 원 : 영남대학교 컴퓨터공학과 박사과정
††† 정 회 원 : 경일대학교 컴퓨터공학부 교수
†††† 종신회원 : 영남대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2008년 1월 26일
수 정 일 : 1차 2008년 8월 26일
심사완료 : 2008년 9월 16일

는 연결 행렬 기반의 select-delelte 알고리즘을 제안하였다[1]. N. Li 등은 k-hop 인접 행렬을 이용하여 행렬 기반의 빠른 네트워크 성능 계산 알고리즘을 제안하였다[2]. 그러나 위의 연구들은 네트워크의 집적도 및 연결 가능성의 성능을 평가하는데 주안점을 두고 있다.

최근 활발히 연구되고 있는 On-demand 방식의 라우팅 프로토콜은 동시에 복수의 연결(Connection)이 이루어지는 애드 혹 네트워크에서 연결이 이루어질 때마다 경로를 구축해야하고 이로 인한 전송 지연이 불가피하다. 따라서 주기적으로 네트워크 내의 모든 노드들에 대한 라우팅 정보를 유지하기 위한 라우팅 오버헤드의 문제를 가지고 있지만, 빠른 경로 설정이 장점인 Table-driven 방식에서 제어패킷의 오버헤드를 줄일 수 있다면 복수의 연결이 이루어지는 애드 혹 네트워크에서 전송 지연을 크게 줄일 수 있다.

본 논문에서는 네트워크의 연결 정보를 나타내는 간단한 인접 행렬을 기반으로 구해진 연결 행렬을 이용한 라우팅 방법을 제안한다. 네트워크 내의 모든 노드들은 네트워크 토폴로지를 나타내는 동일한 인접 행렬 정보를 유지하고 필요할 경우 간단한 행렬 연산을 통해 소스에서 목적지까지의 경로를 결정할 수 있다. 네트워크 토폴로지 정보는 리더노드가 취합하여 인접 행렬의 형태로 네트워크 내의 모든 노드에게 전달한다. 복수의 후보 리더를 이용함으로써 리더 선출 과정의 오버헤드를 줄일 수 있으며, 행렬 연산을 통해 목적지까지의 경로를 역검색 함으로써 빠른 경로 검색이 가능하다.

2. 네트워크의 인접 행렬 모델과 연결 행렬 모델

애드 혹 네트워크의 토폴로지는 무방향 그래프 $G(V, A)$로 나타낼 수 있다. 여기서 V는 네트워크의 노드의 집합을 나타내고 A는 네트워크의 토폴로지를 표시하는 인접 행렬을 나타낸다. N개의 노드를 가진 네트워크에서 인접(1 홉 연결) 행렬 $A=\{a_{ij}\}$에서 (i, j)의 값이 1인 경우는 노드 i에서 노드 j 로의 직접 연결이 있는 경우를 의미하고, (i, j)의 값이 0인 경우에는 노드 i 에서 노드 j 로의 직접 연결이 없는 경우를 나타낸다. 인접 행렬을 이용하여 (i, j)의 값이 노드 i 에서 노드 j 로 연결되는데 필요한 최소 홉 수를 나타내는 연결 행렬 $C=\{c_{ij}\}$를 구할 수 있다[3].

연결 행렬 C 는 다음과 같이 정의할 수 있다.

첫째, 인접 행렬은 노드 사이의 1 홉 연결정보를 나타내기 때문에 식 (1)로 표시할 수 있다.

$$C^{(1)} = A. \tag{1}$$

둘째, 인접 행렬 A를 제곱한 $A^2 = a_{ij}^{(2)}$의 요소들은 식 (2)와 같이 표시된다.

$$a_{ij}^{(2)} = \sum_{k=1}^{N} a_{ik}a_{kj} = \begin{cases} 0, & \text{경로가 없을 때} \\ >0, & \text{최소 하나 이상의 경로가 있을 때.} \end{cases} \tag{2}$$

$a_{ij}^{(2)}$은 아래의 식 (3)과 같이 수정한다.

$$b_{ij}^{(2)} = \begin{cases} 0, & i=j \\ 0, & a_{ij}=1 \\ 2, & a_{ij}^{(2)}>0 \text{ 일 때} (i \neq j \text{ and } a_{ij}=0) \\ 0, & \text{그렇지 않으면.} \end{cases} \tag{3}$$

마지막으로 2 홉 연결 행렬 $c_{ij}^{(2)}$은 식 (4)와 같다.

$$c_{ij}^{(2)} = a_{ij} + b_{ij}^{(2)} = c_{ij}^{(1)} + b_{ij}^{(2)}. \tag{4}$$

연결 행렬을 구하기 위한 일반식은 식 (5)와 같이 나타낼 수 있다.

$$c_{ij}^{(m)} = c_{ij}^{(m-1)} + b_{ij}^{(m)},\ m \geq 2 \tag{5}$$

여기서,

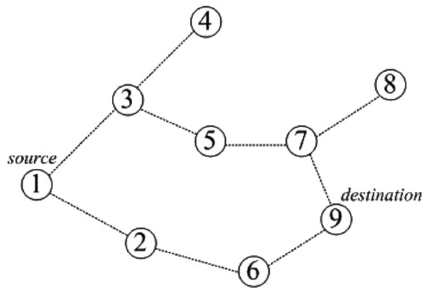
$$b_{ij}^{(m)} = \begin{cases} 0, & i=j \\ 0, & c_{ij}^{(m-1)}>0 \\ m, & \sum_{k=1}^{N} c_{ik}^{(m-1)}a_{kj}>0\ (i \neq j \text{ and } c_{ij}^{(m-1)}=0) \\ 0, & \text{그렇지 않으면.} \end{cases}$$

(그림 1)의 (a)는 네트워크의 토폴로지를 나타내고 (b)는 이에 대응하는 인접 행렬을 보여준다.

(그림 2)의 (a)는 2-홉 행렬을 나타내고 (b)는 행렬 연산에 의해 주어진 최종 연결 행렬을 보여준다.

인접 행렬로부터 연결 행렬을 계산하기 위한 알고리즘은 아래와 같다.

step 1 : $N \times N$의 인접 행렬 A와 두 개의 $N \times N$ 행렬 B, C를 정의한다.



(a)

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

(b)

(그림 1) 네트워크 모델(a) 과 인접 행렬(b)

$$A^{(2)} = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 2 \\ 1 & 2 & 0 & 1 & 1 & 0 & 2 & 0 & 0 \\ 2 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 2 & 0 & 0 & 1 & 2 & 2 \\ 2 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 2 \\ 0 & 2 & 0 & 0 & 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

(a)

$$C = A^{(4)} = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 3 & 4 & 3 \\ 1 & 0 & 2 & 3 & 3 & 1 & 3 & 4 & 2 \\ 1 & 2 & 0 & 1 & 1 & 3 & 2 & 3 & 3 \\ 2 & 3 & 1 & 0 & 2 & 4 & 3 & 4 & 4 \\ 2 & 3 & 1 & 2 & 0 & 3 & 1 & 2 & 2 \\ 2 & 1 & 3 & 4 & 3 & 0 & 2 & 3 & 1 \\ 3 & 3 & 2 & 3 & 1 & 2 & 0 & 1 & 1 \\ 4 & 4 & 3 & 4 & 2 & 3 & 1 & 0 & 2 \\ 3 & 2 & 3 & 4 & 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

(b)

(그림 2) 2-홉 행렬(a) 과 연결 행렬(b)

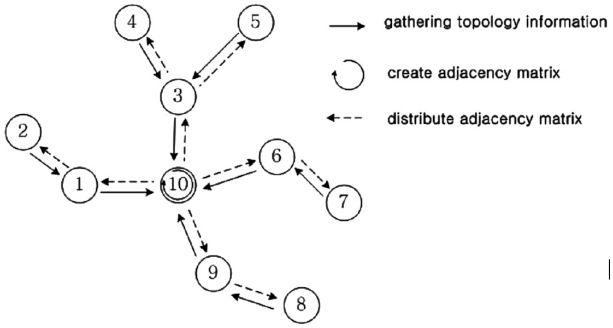
step 2 : *C=A, B=0*으로 초기화한다.

step 3 : For *m=2*에서 가장 긴 홉 거리까지
3-1 : For 모든 노드의 *(i, j) = (1, 1)* to *(N, N)*까지
3-1-1 : 만약 *i=j* 라면 다음 노드로 넘어간다.
3-1-2 : 만약 $c_{ij} > 0$라면 다음 노드로 넘어간다.
3-1-3 : For *k=1* to *N*
3-1-3-1 : 만약 어떤 k에 대해 $c_{ik} > 0$ 그리고 $a_{kj} > 0$이라면
3-1-3-1-1 : $b_{ij} = m$
3-1-3-1-2 : 루프를 벗어나 다음 노드의 *(i, j)*로 간다.
3-2 : *C=C+B, B=0*

step 4 : 계산이 완료되면 연결 행렬 *C*가 구해진다. *C*의 모든 요소의 합을 *N(N-1)*로 나누면 네트워크의 평균 홉 거리가 구해진다.

3. 네트워크 토폴로지 정보의 수집과 리더 선출

본 연구에서는 인접 행렬 생성 및 관리의 효율을 높이고 라우팅의 성능을 높이기 위해 리더 노드가 네트워크의 토폴로지 정보를 나타내는 인접 행렬을 수집하고 이를 네트워크 내의 모든 노드들에게 전달하는 방법을 제안한다. (그림 3)은 리더 노드를 이용하여 토폴로지 정보를 수집하고 이를 인접 행렬로 나타낸 후에 네트워크 내의 노드들에게 전달하는 과정을 보여주고 있다. 이러한 과정을 통해 네트워크 내의 모든 노드들은 토폴로지 정보를 나타내는 동일한 인접 행렬을 가지게 되고[4,5], 각 노드는 필요에 따라 연결 행렬을 이용하여 본 연구에서 제안하는 경로 역검색 방식으로 소스 노드에서 목적지 노드까지 경로가 존재하는지를 쉽게 파악할 수 있다.



(그림 3) 리더에 의한 토폴로지 정보의 수집과 전달

3.1 네트워크 토폴로지 정보 수집

네트워크 내의 모든 노드들은 이웃 노드 정보를 수집하기 위해 주기적으로 hello 메시지를 주고받는다. 시작 노드는 네트워크의 토폴로지 정보를 수집하기 위해 모든 이웃 노드에게 토폴로지 질의 메시지를 전송한다. 토폴로지 질의 메시지를 전송한 노드는 부모 노드가 되고 수신한 노드는 자식 노드가 된다. 토폴로지 질의 메시지를 수신한 노드는 이를 다시 자식 노드들에게 전송한다. 토폴로지 질의 메시지를 받은 노드는 모든 자식 노드로부터 토폴로지 응답 메시지를 받았거나, 대기 시간 안에 아무런 acknowledgement를 받지 않은 경우 혹은 대기 시간이 만료 되었을 경우에 토폴로지 응답 메시지를 자신의 부모 노드에게만 전송한다. 초기의 대기 시간은 식 (6)과 같다.

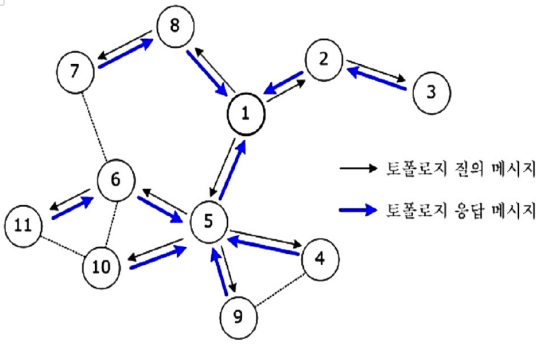
$$wait\ time = 2 \times node\ traversal\ time \tag{6}$$

Acknowledgement 메시지를 받은 후의 대기 시간은 다음 식 (7)와 같다.

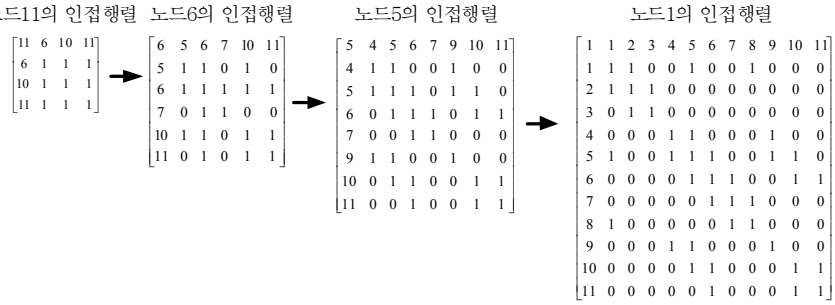
$$wait\ time = \max(wait\ time, node\ traversal\ time \times (network\ diameter - hop\ count)) \tag{7}$$

네트워크의 말단 노드는 default 대기 시간을 기다린 후에 부모 노드에게 토폴로지 응답 메시지를 전달한다. 토폴로지 응답 메시지를 전달할 때는 자신이 가지고 있는 이웃 노드의 정보를 같이 전달하고, 그 때까지 수집된 이웃 노드 정보를 바탕으로 인접 행렬 정보를 만들어 부모 노드에게 전달한다. 이것은 인접 행렬이 역확산되는 과정이라 할 수 있다. (그림 4)는 토폴로지 응답 메시지의 전달 과정과 토폴로지 응답 메시지를 통해 네트워크의 토폴로지를 나타내는 인접 행렬이 역확산되는 과정을 보여준다. (그림 4)의 (a)는 토폴로지 응답 메시지의 전달 과정을 보여주고, (그림 4)의 (b)는 네트워크의 말단 노드인 노드 11부터 토폴로지 응답 메시지를 통해 네트워크의 인접 행렬이 역확산되는 과정을 보여준다.

토폴로지 응답 과정을 거친 후에 시작 노드는 네트워크의 모든 토폴로지 정보를 수집할 수 있고 이를 바탕으로 전체 네트워크의 토폴로지를 나타내는 인접 행렬을 생성한다. 인



(a)



(b)

(그림 4) 토폴로지 응답 메시지의 전달(a)과 인접 행렬의 역확산 과정(b)

접 행렬을 생성한 후 시작 노드는 이를 토폴로지 광고 메시지를 이용하여 네트워크 내의 모든 노드들에게 전달한다. 시작 노드는 리더 노드의 역할을 계속하거나 보다 유리한 노드를 리더 노드로 선정하여 역할을 넘겨줄 수 있다.

3.2 리더 리스트를 이용한 리더 선출

토폴로지 응답 과정을 통해 네트워크의 연결 정보를 수집한 후 시작 노드는 리더 선출의 과정을 수행한다. 본 논문의 리더 선출 과정은 [6]에서 제시한 리더 선출 알고리즘에 기반을 두고 있다. 그러나 애드 혹 네트워크에서는 토폴로지의 변화가 동적이므로 리더가 수시로 바뀔 수 있다. 따라서 리더 선출 과정의 오버헤드를 방지하기 위해 [6]에서 제안하고 있는 각 노드가 단일 리더를 유지하는 방법 대신 본 논문에서는 복수의 후보 리더들로 이루어진 리더 리스트를 사용한다[7]. 각 노드는 5개의 후보 리더로 구성된 리더 리스트를 유지하며 그 중에서 첫 번째 노드가 네트워크의 활성 리더로 간주된다. 일정한 시간 동안 활성 리더가 존재하지 않으면, 새로운 리더 선출의 과정을 거치는 대신 리더 리스트의 두 번째 리더가 활성 리더가 된다. 리더 리스트를 선출한 후 시작 노드는 [7]에서와 같이 leader 메시지를 통해 이를 네트워크의 모든 노드들에게 전달한다.

4. 연결 행렬을 이용한 경로 검색

리더 노드는 네트워크 내의 모든 노드들에게 토폴로지 광고 메시지를 통해 네트워크의 인접 행렬 정보 및 연결 행렬 정보를 브로드캐스트한다. 따라서 패킷의 전송이 필요한 노드는 주어진 두개의 행렬을 통해 목적지 노드로의 경로를 검색할 수 있다. 여기에서는 연결 행렬을 이용한 두 가지의 경로 검색 방법에 대해 설명한다. 첫 번째는 목적지로부터 소스 노드까지의 최단 경로를 검색하는 방법이고, 두 번째는 목적지로부터 소스 노드까지의 노드-비중첩 다중 경로를 검색하는 방법이다.

4.1 순방향 경로 검색과 역방향 경로 검색

각 노드는 인접 행렬만을 이용해서 깊이 우선 탐색(DFS, Depth First Search) 알고리즘이나 넓이 우선 탐색(BFS, Breadth First Search) 알고리즘을 적용하여 소스 노드로부터 목적지 노드까지의 경로를 검색할 수 있다[8]. 그러나 본 논문에서는 연결 행렬 기반의 경로 역검색(MBBS, Matrix-Based Path Backward Search) 방법을 제안한다[9]. 이 방법은 소스 노드가 목적지 노드까지의 경로를 검색할 때, 연결 행렬을 이용하여 목적지 노드로부터 소스 노드까지 역으로 경로를 검색하는 방법이다.

소스 노드로부터 목적지까지의 홉 거리가 k 홉일 경우 인접 행렬을 이용하여 순방향으로 경로를 설정하기 위해 BFS 알고리즘으로 검색해야 하는 노드의 수는 식 (8)과 같다.

$$\text{순방향 총 검색 노드 수(BFS)} = \sum_{i=1}^{k} N_{i\_adj} \qquad (8)$$

여기서 k는 목적지까지의 홉 수, N_{i_adj}는 각 노드에서 1홉

거리에 있는 이웃 노드의 수이다. 즉 소스 노드로부터 목적지까지의 홉 거리에 있는 모든 노드를 검색해야 경로를 설정할 수 있다. 연결 행렬을 이용하는 역방향 경로 검색의 경우 검색해야 하는 노드의 수는 식 (9)와 같이 나타낼 수 있다.

$$\text{역방향 총 검색 노드 수(MBBS)} = \sum_{i=1}^{k} (N_{i-1\_connected} \in N_{i\_adj}) \quad (9)$$

여기서 $N_{i-1_connected}$는 i번째 홉의 노드와 1 홉으로 연결된 소스 쪽 노드의 수이다. 즉 기준인 i번째 홉의 노드보다 1 홉 적은 노드로, 목적지 노드로부터 소스로 직접 연결된 노드만을 검색함으로써 경로 검색의 과정을 줄일 수 있다.

4.2 연결 행렬을 이용한 최단 경로 역검색

네트워크의 인접 행렬을 이용한 행렬 연산을 통해 네트워크 내의 모든 노드들의 연결 행렬을 구할 수 있지만 목적지까지의 최단 경로를 역검색 할 때는 행렬 연산의 복잡성을 줄이기 위해 소스 노드에서 목적지 노드까지의 홉 정보를 알 때까지만 연결 행렬을 계산한다. 역경로 검색을 위해 본 논문에서는 소스 노드로부터 목적지 노드까지 홉 수를 나타내는 홉 리스트를 사용한다. 각 홉 리스트에는 소스 노드로부터 같은 홉 수를 가진 노드들이 속하게 된다. 소스 노드로부터 목적지까지의 홉 수가 k라면 k-1 홉 리스트에서 목적지 노드와 연결된 노드를 검색한다. 만약 k-1 홉 리스트 중에 목적지 노드의 이웃 노드가 있다면, k-2 홉 리스트 중에서 그 노드와 연결된 노드를 검색한다. 이러한 과정을 소스 노드가 발견될 때까지 계속한다.

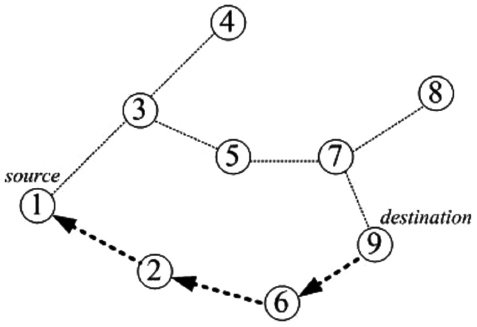
(그림 5)의 (a)와 같은 네트워크에서 소스 노드에서 목적지까지의 홉 수가 3 홉이기 때문에 (그림 5)의 (b)와 같은 3 홉 연결 행렬을 통해 목적지까지의 최단 경로를 역 검색할 수 있다. (그림 5)의 (c)는 소스 노드로부터 3 홉까지의 모든 노드들을 홉 수 별로 나타낸 홉 리스트이다. (그림 5)에서 목적지인 노드 9는 소스 노드인 노드 1로부터 3 홉 거리에 있다. 본 논문에서 제안한 방법에 의해 먼저 2 홉 리스트 중에서 목적지인 노드 9의 이웃 노드를 검색한다. 2 홉 리스트 중에서 노드 6이 목적지 노드의 이웃 노드이다. 다음으로 1 홉 리스트 중에서 노드 6의 이웃 노드를 검색하면 노드 2가 노드 6의 이웃 노드이다. 따라서 소스 노드로부터 목적지까지 1-2-6-9의 최단 경로를 검색할 수 있다. 노드 7의 경우 비록 목적지 노드의 이웃 노드이지만 소스 노드로부터 3 홉 떨어져 있기 때문에 경로 검색에서 제외된다.

연결 행렬을 이용한 최단 경로 역검색의 알고리즘은 다음과 같다.

step 1 : 목적지까지의 홉 수를 알 때까지 연결 행렬 C를 계산한다. 그리고 목적지까지의 홉 수 k를 입력한다.

step 2 : 소스 노드로부터의 모든 n-홉 노드들을 홉 리스트에 저장한다. (n = 1, 2, …, k-1);

step 3 : While (k ≠ 0)



(a)

$$C = A^{(3)} = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 3 & 0 & 3 \\ 1 & 0 & 2 & 3 & 3 & 1 & 3 & 0 & 2 \\ 1 & 2 & 0 & 1 & 1 & 3 & 2 & 3 & 3 \\ 2 & 3 & 1 & 0 & 2 & 0 & 3 & 0 & 0 \\ 2 & 3 & 1 & 2 & 0 & 3 & 1 & 2 & 2 \\ 2 & 1 & 3 & 0 & 3 & 0 & 2 & 3 & 1 \\ 3 & 3 & 2 & 3 & 1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 3 & 0 & 2 & 3 & 1 & 0 & 2 \\ 3 & 2 & 3 & 0 & 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

(b)

source	1-hop	2-hop	3-hop
①	2, 3	4, 5, 6	7, ⑨

(c)

(그림 5) 연결 행렬을 이용한 최단 경로 역검색

3-1 : k-1 홉 리스트의 노드 중에서 목적지의 이웃 노드를 선택한다.
3-2 : 이 노드를 경로 셋에 저장한다.
3-3 : k = k-1;
3-4 : 이 이웃 노드가 새로운 목적지가 된다.

step 4 : 경로 셋을 저장한다.

4.3 연결 행렬을 이용한 노드-비중첩 다중 경로 역검색

인접 행렬의 행렬 연산에 의한 연결 행렬의 경로 역검색 방법을 이용하면 목적지로부터 소스 노드까지의 최단 경로뿐 아니라 노드-비중첩 다중 경로를 검색할 수 있다. 노드-비중첩 다중 경로를 역검색하기 위해서는 최단 경로를 검색하는 방법과는 달리 네트워크의 가장 긴 홉을 알 때까지 연결 행렬을 계산해야 한다. 본 논문에서는 연결 행렬을 이용한 노드-비중첩 다중 경로의 역검색을 위해 소스 노드에서 가장 긴 홉을 가진 노드까지 모든 노드들을 홉 수 별로 나타낸 다중 경로 홉 리스트를 사용한다. 먼저 연결 행렬에서 목적지의 이웃 노드들을 검색한다. 소스 노드로부터 목적지까지의 홉 수가 k 일때 목적지의 이웃 노드의 홉 수는 k-1, k, k+1 중의 하나가 된다. 목적지 이웃 노드의 홉 수가 k-1일 경우에는 앞의 절에서 설명한 방법으로 목적지까지의 경로를 검색하면 목적지 노드까지의 최단 경로가 검색된다. 목적지 이웃 노드의 홉 수가 k나 k+1일 경우에는 k-1 홉

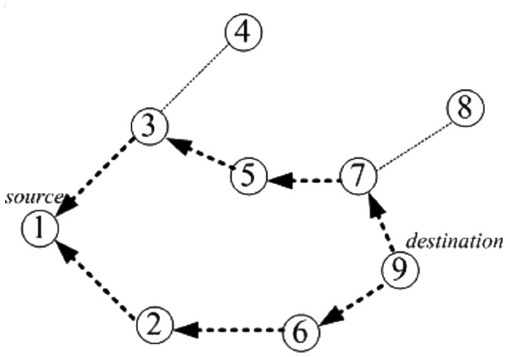
리스트나 k 홉 리스트의 노드 중에서 경로 설정에 사용되지 않은 이웃 노드가 있는지를 검색한다. 경로 설정에 사용되지 않은 이웃 노드가 있다면 앞에서 설명한 방법과 같이 1 홉을 감소하면서 이웃 노드들을 검색한다. 만약 경로에 사용되지 않은 이웃 노드가 없으면 k+1, k+2 홉 리스트 노드들 중에서 연결된 이웃 노드가 있는지를 검색해서 네트워크의 가장 긴 홉 리스트까지 검색한다. 이러한 과정을 소스 노드가 검색될 때까지 반복한다.

연결 행렬을 이용한 노드-비중첩 다중 경로의 역검색 알고리즘은 다음과 같다.

step 1 : 연결 행렬 C를 계산한다. 목적지의 홉 k를 입력한다.
step 2 : 소스 노드로부터의 모든 n-홉 노드들을 홉-셋 리스트에 저장한다. (n = 1, 2, …, 가장 긴 홉 거리);
step 3 : While (k ≠0)
3-1 : k-1 홉 셋 노드 중에서 목적지의 이웃 노드를 선택한다.
3-1-1 : 만약 경로 설정에 사용되지 않은 이웃 노드가 있으면 k=k-1.
3-1-2 : 만약 경로 설정에 사용되지 않은 이웃 노드가 없으면 k=k+1.
3-2 : 그 노드를 경로 셋에 저장한다.
3-3 : 이 이웃 노드가 새로운 목적지 노드가 된다.
step 4 : 경로 셋을 출력한다.

(그림 6)은 연결 행렬을 이용한 노드-비중첩 다중 경로의 역검색 과정을 보여주고 있다.

(그림 6)의 (b)는 네트워크의 모든 노드들의 홉 수를 계산한 4 홉 연결 행렬이다. (그림 6)의 (c)에서 목적지의 이웃 노드는 노드 6과 노드 7이다. 먼저 이전에서 설명한 방법으로 노드 6을 이용하여 소스 노드까지 최단 경로를 검색한다. 다음으로 노드 7을 경유하는 노드-비중첩 다중 경로를 검색하는데 노드 7보다 1 홉이 작은 2 홉 리스트의 노드 중에서 노드 7의 이웃 노드를 검색한다. 2 홉 리스트의 노드 중에서 노드 5가 다른 경로에 사용되지 않은 노드 7의 이웃 노드이다. 1 홉 리스트의 노드 중에서는 노드 3이 노드 5의 이웃 노드이다. 그 결과 1-2-6-9, 1-3-5-7-9의 노드-비중첩 다중 경로가 검색된다.



(a)

$$C = A^{(4)} = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 3 & 4 & 3 \\ 1 & 0 & 2 & 3 & 3 & 1 & 3 & 4 & 2 \\ 1 & 2 & 0 & 1 & 1 & 3 & 2 & 3 & 3 \\ 2 & 3 & 1 & 0 & 2 & 4 & 3 & 4 & 4 \\ 2 & 3 & 1 & 2 & 0 & 3 & 1 & 2 & 2 \\ 2 & 1 & 3 & 4 & 3 & 0 & 2 & 3 & 1 \\ 3 & 3 & 2 & 3 & 1 & 2 & 0 & 1 & 1 \\ 4 & 4 & 3 & 4 & 2 & 3 & 1 & 0 & 2 \\ 3 & 2 & 3 & 4 & 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

(b)

source	1-hop	2-hop	3-hop	4-hop
①	2, 3	4, 5, 6	7, ⑨	8

(c)

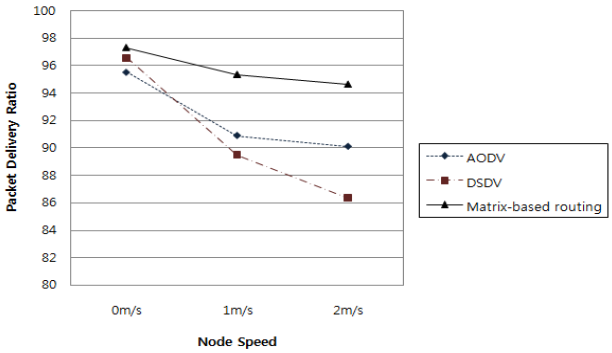
(그림 6) 연결 행렬을 이용한 노드-비중첩 다중 경로 역검색

5. 성능 평가

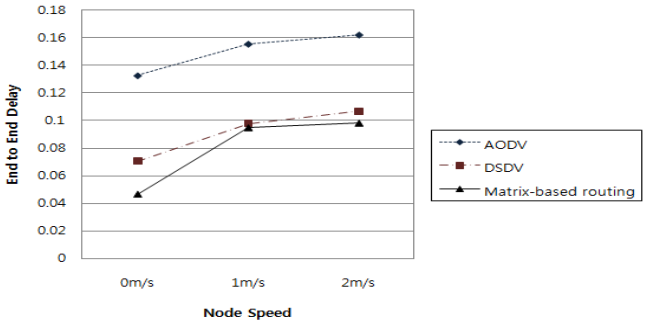
지금까지의 애드 혹 네트워크에서의 성능 평가는 대부분 네트워크 내에 하나의 연결만이 있다고 가정하고 이루어졌다. 그러나 실제적으로 애드 혹 네트워크에서는 동시에 여러 개의 연결이 이루어지고 있다. 이러한 경우 주기적으로 네트워크의 연결 정보를 수집하는 Table-driven 방식이나 각각의 연결이 이루어질 때 마다 경로를 설정해야 하는 On-demand 방식에 비해 본 논문에서 제안하는 인접 행렬 기반의 라우팅 방식은 뛰어난 성능을 보일 수 있다. 애드 혹 네트워크에서 동시에 여러 개의 연결이 존재하는 경우에 인접 행렬을 이용한 라우팅의 성능을 평가하기 위하여 기존의 Table-driven 방식의 대표적인 라우팅 프로토콜인 DSDV (Destination-Sequence Distance-Vector), On-demand 방식의 대표적인 라우팅 프로토콜인 AODV(Ad hoc On-demand Distance-Vector)와 성능을 비교하였다.

시뮬레이션은 네트워크 시뮬레이터인 NS2[10]를 사용했으며, 버전은 ns-allinone-2.30이다. 본 실험은 1000m×1000m의 네트워크 영역에 33개의 노드를 배치하였고 전송 범위는 250m로 하였다. 동시 최대 연결 수는 4개로 하였으며 총 연결의 수는 100개로 하였다. 노드의 이동 속도가 각각 0m/sec, 1m/sec, 2m/sec 일 때 패킷 전달율, 전송 지연율, 라우팅 오버헤드를 비교하였다. 실험 시간은 500초이며 실험 회수는 20회를 반복하여 그 평균을 구하였다.

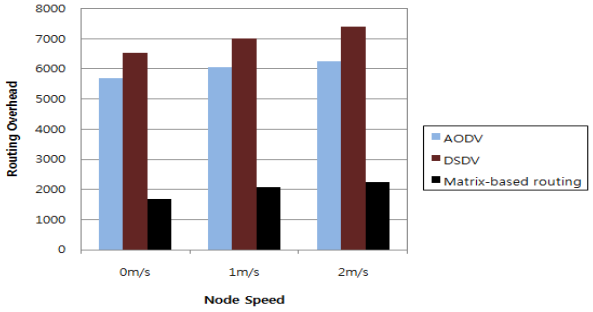
(그림 7)은 노드의 이동 속도에 따른 패킷 전달율을 나타낸다. 노드가 고정 상태일 때의 패킷 전달율은 인접 행렬 기반의 라우팅 방식과 DSDV가 거의 같은 성능을 보이고 있다. AODV는 각 연결마다 경로 설정 과정을 거치기 때문에 가장 낮은 성능을 보인다. 그러나 노드의 이동 속도가 빨라짐에 따라 DSDV는 패킷 전달율이 급격히 감소한 반면 인접 행렬 기반의 라우팅 방식은 여전히 높은 패킷 전달율을 가지고 있다. 이는 DSDV는 주기적으로 네트워크의 연결 정보를 수집하기 때문에 노드의 이동 속도가 빠른 경우 네



(그림 7) 패킷 전송율



(그림 8) 전송 지연율



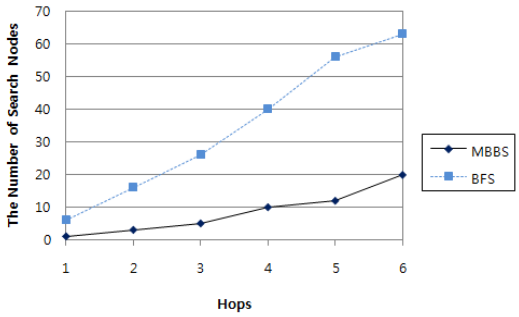
(그림 9) 라우팅 오버헤드

트워크의 연결 정보를 충분히 나타내지 못하기 때문이다.

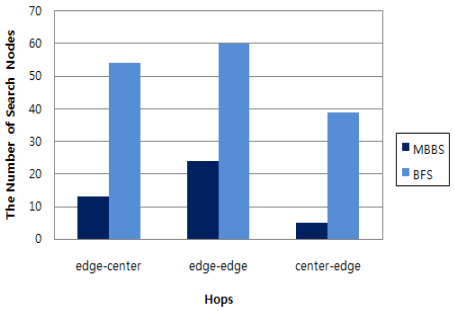
(그림 8)은 노드의 이동 속도에 따른 전송 지연율을 나타낸다. AODV는 각 연결 마다 경로 설정 과정을 거치기 때문에 연결의 수가 많아질수록 전송 지연이 증가하는 반면 DSDV와 인접 행렬 기반의 라우팅 방식은 네트워크 내의 모든 노드들에 대한 연결 정보를 가지고 있기 때문에 연결의 수가 많아지더라도 전송 지연이 발생하지 않는다. 노드의 이동 속도가 빠른 경우에도 인접 행렬 기반의 라우팅 방식은 가장 뛰어난 성능을 보이고 있다.

(그림 9)는 노드의 이동 속도에 따른 라우팅 오버헤드를 나타낸다. 고정 상태에서 뿐만 아니라 노드의 이동 속도가 빠른 경우에도 인접 행렬 기반의 라우팅 방식은 가장 뛰어난 성능을 보이고 있다.

다음으로 본 논문에서 제안한 연결 행렬을 이용한 경로 역검색 방법(MBBS)의 성능을 넓이 우선 탐색 방법(BFS)과 비교하였다. 실험 영역은 1000 x 1000m에 64개의 노드를 임



(그림 10) 목적지의 홉 수에 따른 검색 노드 수



(그림 11) 소스 노드와 목적지 노드의 위치에 따른 검색 노드 수

의로 배치하였다. 신호의 전송 범위는 250m로 하였으며 노드는 이동하지 않은 고정 상태로 하였다. 연결 행렬을 얻기 위한 행렬 연산의 복잡도는 고려하지 않았다. 경로 설정을 위해 검색하는 노드의 수를 알아보기 위해 목적지의 홉 거리를 변화 시켰으며 소스와 목적지의 위치에 따른 검색 노드의 수를 비교하였다. 실험은 노드의 배치를 달리하여 20회 실시하여 그 평균을 구하였다.

(그림 10)은 목적지의 홉 거리에 따라 경로 설정을 위해 검색하는 노드의 수이다. 목적지의 홉 수가 6 홉일 경우, BFS 알고리즘은 경로 설정을 위해 네트워크의 거의 모든 노드를 검색하는데 비해 본 논문에서 제안한 경로 역검색 알고리즘은 20개의 노드를 검색한 후 경로 설정을 할 수 있다. 행렬 연산의 복잡도를 고려하지 않는다면 목적지의 홉 거리가 멀어질수록 본 논문에서 제안한 연결 행렬을 이용한 경로 역검색 방법의 성능이 뛰어나다는 것을 알 수 있다. 동일한 노드 수를 가진 실험 영역 내에서 목적지의 홉 수가 많을 때 뛰어난 성능을 보인다는 것은 네트워크 내의 노드의 수가 증가할수록 경로 역검색 방법이 뛰어난 효율을 보인다고 할 수 있다.

(그림 11)은 소스 노드와 목적지 노드의 위치에 따른 경로 검색 노드 수이다. 정방형의 네트워크에서 소스 노드와 목적지 노드는 각기 네트워크의 중앙이나 가장 자리에 위치할 수 있다. 소스 노드로부터 경로 검색을 하는 넓이 우선 탐색의 경우 소스 노드가 네트워크의 중앙에 위치하는 경우 검색해야 하는 이웃 노드의 수가 증가한다. 그러나 본 논문에서 제안한 연결 행렬 기반의 경로 역검색 방법은 소스 노드가 네트워크의 중앙에 위치하고 목적지가 네트워크의 가장자리에 위치할 때 가장 좋은 성능을 보이고 있다.

6. 결론 및 향후 과제

애드 혹 네트워크에서 전송 지연에 민감한 실시간 어플리케이션을 위해서는 효율적인 경로 설정이 필요하다. 효율적인 경로 설정을 위해 본 논문에서는 네트워크의 토폴로지 정보를 인접 행렬로 나타내고, 네트워크의 토폴로지를 유지 관리하기 위해 복수의 리더를 선출함으로써 리더 선출 과정의 오버헤드를 감소시켰다. 또한 경로 설정이 필요한 경우 행렬 연산을 통해 네트워크의 연결 행렬을 계산하여 목적지까지의 경로를 역검색 하는 방법을 제안하였다. 실험 결과 애드 혹 네트워크에서 동시에 연결의 수가 많을수록 본 논문에서 제안한 인접 행렬 기반의 라우팅 방식이 뛰어난 성능을 보여주고 있음을 알 수 있었다.

그러나 본 논문에서 제안한 인접 행렬을 이용한 경로 역검색의 방법에서는 인접 행렬의 연산을 위한 복잡도를 고려하지 않았다. 따라서 향후 연구에서는 인접 행렬 연산의 복잡도를 줄이기 위한 알고리즘의 개발이 필요하다.

참 고 문 헌

[1] J. Zhang and W. K. G. Seah, “Topology-based Capacity Analysis for Ad Hoc Networks with End-to-End Delay Constraints,” Proc. of IEEE 6th CAS Symposium on Emerging Technologies: Mobile and Wireless Communications, Shanghai, China, pp.541-544, 2004.

[2] Ning Li, Yan Guo, Shaoren Zheng, Chang Tian and Jun Zheng, “A Matrix-Based Fast Calculation Algorithm for Estimating Network Capacity of MANETs,” ICW/ICHSN/ICMCS/SENET pp.407-412, 2005.

[3] L. E. Miller, “Multihop connectivity of arbitrary networks,” http://w3.antd.nist.gov/wctg/netanal/ConCalc.pdf, March, 2001.

[4] HangKon Kim, SungSoo Lee and ChongGun Kim, “Design of Knowledge Discovery Agent for a Bit-Map on Ad Hoc Mobile Networks,” KES-AMSTA 2007, pp.738-746, 2007.

[5] SungSoo Lee, HangKon Kim and ChongGun Kim, “A Knowledge Discovery Agent for a Topology Bit-map in Ad Hoc Mobile Networks,” Journal of Universal Computer Science, Vol.14, No.7, pp.1105-1117, http://www.jucs.org/jucs_14_7/a_knowledge_discovery_agent, 2008.

[6] Vasudevan, S., Kurose, J and Towsley, D., “Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks”. Proceedings of the 12th IEEE International Conference on Network Protocols(ICNP), pp.350-360, 2004.

[7] Sungsoo Lee, Rahman M. Muhammad and Chonggun Kim, “A Leader Election Algorithm within candidates on Ad Hoc Mobile Networks,” International Conference on Embedded Software and Systems (ICESS-07), LNCS4523, 2007.

[8] Weiss. M. A, “Data Structures and Algorithm Analysis,” 2nd edition, Addison Wesley Longman, 1997.

[9] Chonggun Kim, Elmurod Talipov and Byoungchul Ahn, “A Reverse AODV Routing Protocol in Ad Hoc Mobile Networks,” EUC Workshops 2006, LNCS 4097, pp.522- 531, 2006.

[10] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/, 2006.

이 성 수

e-mail : lssung@chol.com

1985년 경북대학교 사회학과(학사)

2004년 영남대학교 대학원 컴퓨터정보통신학과(공학석사)

2008년 영남대학교 대학원 컴퓨터공학과(공학박사)

현 재 진량고등학교 교사

관심분야 : Mobile Ad hoc Network, Ubiquitous Sensor Network, Network Management

김 정 미

e-mail : tkvkdldj@ynu.ac.kr

2004년 한국방송통신대학교 컴퓨터과학과(학사)

2006년~현 재 영남대학교 컴퓨터공학과 박사과정

관심분야 : 컴퓨터 네트워크, 무선 모바일 네트워크, 분산처리

박 희 주

e-mail : hjpark@kiu.ac.kr

1978년 영남대학교 전자공학과(공학사)

1981년 영남대학교 대학원 전자공학과(공학석사)

1995년 대구가톨릭대학교 대학원 전산통계학과(이학박사)

1982년~현 재 경일대학교 컴퓨터공학부 교수

관심분야 : 신경회로망, 패턴인식

김 종 근

e-mail : cgkim@yu.ac.kr

1981년 영남대학교 전자공학과(학사)

1987년 영남대학교 전자공학과(석사)

1991년 (일본) 전기통신대학 박사

1997년 (미국) Virginaia Tech. 연구교수

2003년 (미국) UCSC 연구교수

현 재 영남대학교 컴퓨터공학전공 교수

관심분야 : 컴퓨터 네트워크, 무선 모바일 네트워크, 분산처리, 운영체제, 멀티미디어기반 가상강의 시스템