무선 Ad-hoc 네트워크 환경에서 강건한 신장 트리를 유지하는 기법

정회원 강 용 혁*, 엄 영 익*

Maintaining Robust Spanning Tree in Wireless Ad-hoc Network Environments

Yong-Hyeog Kang*, Young Ik Eom* *Regular Members*

요 약

무선 ad-hoc 네트워크는 어떠한 하부 구조와 중앙 관리의 도움 없이 임시 네트워크를 구성하는 무선 이동 호스트들의 집합이다. 무선 ad-hoc 네트워크는 빠르게 배치가능하고 변화에 탄력이 있어야 하는 네트워크 환경에 아주 유용하게 이용된다. 이러한 환경에서 분산 응용을 수행하기 위해서는 어느 정도 오류를 감내할 수 있는 강건한 구조가 요구되며, 또한 응용을 수행하는 데 사용되는 자원의 효율성도 요구된다. 본 논문에서는 무선 ad-hoc 네트워크 환경에서 강건하고 효율적으로 분산 응용이 수행될 수 있도록 강건한 신장 트리를 구성하고 유지하는 기법을 제안한다. 본 논문에서 제안하는 기법은 네트워크 위상의 변화에 영향을 적게 받는 강건한 신장 트리를 구성하기 위해 각 노드들간의 경로 정보를 이용하여 가장 결합도가 높은 경로를 우선적으로 선택하는 방식을 사용한다. 또한, 신장 트리를 효율적으로 유지하기 위해 네트워크 위상의 변화로 인해 신장 트리의 일부분의 붕괴되었을 때 처음부터 재구성하는 방식보다는 기존 신장 트리 정보를 이용하는 방식을 사용한다.

ABSTRACT

A wireless ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any centralized adminstration or standard support services. Wireless ad-hoc networks may be quite useful in that they can be instantly deployable and resilient to change. In this environment, for many crucial distributed applications, it is necessary to design robust virtual infrastructures that are fault-tolerant, self-stabilized, and resource-efficient. For this task, this paper proposes a scheme of maintaining robust spanning trees which are little affected by topological changes. By maintaining such a spanning tree and adapting it to the environments with frequent topological changes, one can improve the reliability and efficiency of many applications that use the spanning tree.

I. 서 론

무선 ad-hoc 네트워크는 어떠한 하부 구조와 중앙 관리의 도움 없이 임시 네트워크를 구성하는 무선 이동 호스트들의 집합이다. 이러한 무선 ad-hoc 네트워크 환경에서는 네트워크를 구성하는 이동 호스트의 이동성으로 인해 네트워크의 위상이 예측할 수 없을 만큼 무작위 적이고 급속하게 변경되는 특색을 가지고 있다. 이러한 환경에서 이동 호스트들간의 통신은 무선 링크를 통해 이루어지므로, 통신 대역폭이 유선 링크에 비해 상당히 작으며, 주변환경에 따라 대역폭의 크기가 변경될 수 있는 특징도 가지고 있다. 또한, 이러한 환경에서의 이동 호스트들은 에너지가 상당히 제한되어 있어서 에너지 소

* 성균관대학교 정보통신공학부 분산시스템연구실 ({yhkang1, yieom}@ece.skku.ac.kr)
논문번호 : 010380-1208, 접수일자 : 2001년 12월 8일

비를 절약하는 기법도 시스템 설계의 중요한 요소로 고려된다.

무선 ad-hoc 네트워크 환경에서 인접하는 무선 이동 호스트들간에는 직접적인 통신이 가능하지만, 직접 통신할 수 없는 호스트로 패킷을 전송하기 위해서는 목적지 노드까지의 경로를 찾은 후에 패킷을 전송해야 한다. 게다가 이동 호스트의 이동성으로 인해 목적지까지의 경로는 고정적인 것이 아니라 동적으로 변경될 수 있으므로 기존의 유선 망에서 사용되는 경로 설정 프로토콜(routing protocol)은 무선 ad-hoc 네트워크에 적용하기가 적합하지 않다. 이러한 무선 ad-hoc 네트워크에 대한 경로 설정 프로토콜에 대한 연구는 IETF의 MANET WG에서 경로 설정 프로토콜의 연구와 개발을 주요 연구 분야로 지정하여 활발하게 진행되고 있다. 대표적인 무선 ad-hoc 환경에서의 경로 설정 프로토콜로는 DSDV, TORA, DSR, 그리고 AODV가 있다[1].

무선 ad-hoc 네트워크의 응용 분야는 그 특성상 유선 망과 연동할 수 없는 곳에서 빠르게 설치할 수 있는 장점을 가지고 있으므로, 군사 전술상의 용도에서부터 전시나 회의, 판매 등을 위한 상업용과 가상교실을 운영하는 교육용 등 다양한 분야에 적용될 수 있다. 또한, 외부 도움 없이 자치적으로 동작하는 이동 네트워크를 구성하여 셀 기반 이동 네트워크(cell-based mobile network)보다 더 안정적이고 설치비용이 싼 임시 네트워크를 구성하는 것과, 기존 네트워크와의 연동을 통해 이동 네트워크 구조를 확장하는 방안으로 응용할 수 있다. 재난이나 소방사고 및 안전 사고와 구조 활동처럼 효율적이고 동적이며 극도로 빠른 배치가 필요한 네트워크도 주요한 응용분야가 될 수 있다.

전통적으로 분산 알고리즘과 프로토콜의 설계는 기반 네트워크가 정적인 호스트로 구성되어 있는 구조를 전제로 하여 연구되었다. 이동 컴퓨팅 환경에서는 이동 호스트의 여러 가지 특성으로 인해 분산 알고리즘에 새로운 문제점들을 가져왔다[2]. 우선 첫 번째 문제점으로 이동 호스트와 함께 분산 알고리즘을 운용하기 위해서는 먼저 이동 호스트의 위치를 찾아야 한다는 문제점이다. 두 번째 문제점으로는 이동 호스트의 이동성으로 인해 분산 알고리즘이 이용하는 네트워크의 논리적 구조가 고정되어 있지 않고 변경된다는 문제점이다. 세 번째 문제점으로는 이동 호스트는 자원이 제약적이기 때문에 필요에 따라 도즈 모드(doze mode)로 동작하거나, 비접속(disconnection) 상태로 동작할 수 있다는 문제점이다. 이러한 도즈 모드나 비접속 상태는 기존의 오류(failure)와는 다른 것으로 간주 되야 한다. 본 논문에서 고려하는 무선 ad-hoc 네트워크 환경에서는 이러한 이동 컴퓨팅 환경에서의 문제점 외에도 네트워크의 위상이 상당히 자주 예상할 수 없을 정도로 변경되는 특성으로 인해 분산 알고리즘을 개발하는 것은 더더욱 어렵다.

본 논문에서는 분산 알고리즘 중에서 여러 응용에서 유용하게 사용되는 신장 트리를 무선 ad-hoc 네트워크 환경에서 적합하도록 관리하는 알고리즘을 제안한다. 본 논문에서는 제안하는 기법의 기본 개념은 무선 ad-hoc 네트워크 환경에서 네트워크 위상의 변화에 영향을 적게 받는 강건한 신장 트리(robust spanning tree)를 구성하고, 효율적으로 유지하는 것이다. 본 논문의 구성은 2장에서 관련 연구에 기술하고, 3장에서는 본 논문의 제안하는 기법을 설명한다. 4장에서는 성능 평가 및 성능 분석을 하며 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

무방향 연결 그래프(connected undirected graph)에서 최소 신장 트리(minimum spanning tree)를 구하는 문제는 그래프에서 나올 수 있는 신장 트리 중에서 신장 트리에 포함되는 링크의 가중치의 총합이 가장 적은 신장 트리를 구하는 문제이다. 이러한 최소 신장 트리는 많은 분산 응용에서 유용하게 사용되는 구조 중에 하나이다. 대표적인 신장 트리의 응용 분야로는 신장 트리를 이용한 리더 선출과 신장 트리를 이용한 방송(broadcast)이 있다. 대부분의 최소 신장 트리를 구하는 분산 알고리즘은 설명을 쉽게 하기 위해 전제 조건으로 링크의 가중치가 모두 다르다는 것과 링크의 방향성이 없다는 것을 전제로 한다. 본 논문에서도 알고리즘의 설명을 쉽게 하기 위해 두 가지 전제 조건을 가정하고 제안 기법을 기술한다. 그러나, 링크의 가중치가 모두 다르지 않는 환경에서도 여러 가지 다른 기준으로 링크의 가중치를 유일하게 만들 수 있다[3]. 또한, 무선 ad-hoc 네트워크처럼 링크의 방향성이 존재하고 링크의 방향에 따라 다른 가중치를 가지는 경우에도 최소 신장 트리를 구할 수 있는 기법이 존재한다[4].

비동기적인 동적인 네트워크 (dynamic asynchronous network)에서는 링크의 생성과 삭제로 인해 네트워크의 위상이 동적으로 변하며, 실제적인 네트워크와 상당히 비슷한 네트워크 모델이다[5]. 정적인

네트워크(static network) 환경에서 개발된 분산 알고리즘은 링크 오류(failure)와 같은 문제가 발생했을 때 해결하는 기법으로 재설정(reset) 프로시저(procedure)를 두어서 처음부터 다시 계산하는 방법을 사용한다. 동적인 네트워크에서 위상이 변경될 때마다 이러한 방법을 똑같이 적용한다면 위상이 자주 변경되는 환경에서는 알고리즘의 오버헤드가 크게 증가하게 되는 문제점이 발생한다[6]. 네트워크에서 노드의 수가 V이고 링크의 개수가 E인 그래프에서 최소 신장 트리를 구하는 분산 알고리즘에 이러한 방법을 사용하여 위상이 변경에 적응하려 한다면 위상이 변경될 때마다 최소 $\Omega(E)$이 통신 복잡도(communication complexity)를 필요로 하게 된다[6]. 네트워크의 위상 변화에 대한 히스토리를 유지하는 기법을 사용하더라도 최소 신장 트리를 구하는 알고리즘은 $\Omega(V)$이 부가적인(amortized) 통신 복잡도를 필요로 하게 된다[5].

무선 ad-hoc 네트워크는 비동기적인 동적인 네트워크와 비슷하다. 이러한 네트워크는 여러 종류의 오류들이 많이 있으므로 분산 알고리즘은 자기안정화(self-stabilization) 개념이 필요하다. 자기안정화란 시스템의 오류 상태에서 복구할 수 있는 능력이다. Dijkstra에 의해 1974년 처음 소개된 자기안정화 개념은 현재는 고장 감내(fault-tolerance)에 대한 접근 방식으로 연구가 계속적으로 진행되고 있다[7,8]. 무선 ad-hoc 네트워크 환경에서도 자기 안정화되는 신장 트리에 대한 연구가 있었다[9]. 그러나, 이 기법 역시 오류 문제에 대한 해결 기법으로 재설정 기법을 썼으며, 재구성 기법으로 인해 네트워크의 직경을 D라 할 때 자기안정화 시간 복잡도(time complexity)는 $O(D)$를 필요로 한다. 그러나 한가지 오류에 대해 전역적인 알고리즘을 수행함으로서 자원의 중요한 무선 ad-hoc 네트워크의 메시지 통신량을 증가시키는 문제점을 야기한다.

무선 ad-hoc 네트워크 환경에서도 분산 알고리즘에 대한 연구가 진행되고 있다[10,11,12,13]. 분산 알고리즘에서 신장 트리와 관련된 연구는 신장 트리를 구성하고 관리하는 기법보다는 신장 트리를 이용하는 응용에 초점이 맞추어져 있다[10,11]. 한 논문에서는 네트워크의 위상을 자주 변화하는 곳과 덜 변화하는 곳으로 구분하는 기법을 사용하였다[10]. 이 기법은 네트워크 위상이 덜 변화하는 곳에서는 신장 트리를 구성하여 구성된 신장 트리를 이용하고, 네트워크 위상이 자주 변화하는 곳은 플러딩(flooding) 기법을 이용하여 위상에 적응성 있는 분산 알고리즘을 제안하였다. 이 기법에서 사용되는 신장 트리는 각각의 노드마다 다르게 나타날 수 있으므로 일반적인 분산 응용에 적합하지 않는 신장 트리이다. 또 다른 논문에서는 자기안정화 개념을 도입하여 위상의 변화를 다루는 분산 오류 감내 알고리즘(distributed fault tolerant algorithm)을 시도했다[11]. 그러나 이 논문의 최소 신장 트리는 멀티캐스트 트리를 위한 트리로 사용되어 멀티 캐스트 소스를 최상위로 하는 계층적인 구조를 가지므로 일반적인 신장 트리에 대한 연구가 아니다.

무선 ad-hoc 네트워크 환경에서 라우팅을 위해 신장 트리 외에도 지배 집합(dominating set)이나 단일 그래프(unit graph)를 구하는 분산 알고리즘에 대한 연구가 있었다[12,13]. 한 논문에서는 상대적으로 안정된(stable) 노드들의 부분 집합이 있다고 가정하고 그 부분 집합으로부터 백본(backbone)을 구성하였다[12]. 또한, 구성된 백본이 최소 연결 지배 집합의 성질을 만족하면 신뢰할 수 있는 멀티캐스팅(multicasting)과 브로트캐스팅(broadcasting)이 가능하다고 기술하였다. 그러나 백본을 구성하는 노드가 단절되었을 경우에는 이러한 문제점을 복구하는데 오랜 시간이 소요된다는 것을 성능 평가 결과에서 언급하였다. 또다른 논문에서는 단일 그래프를 이용하여 패킷의 전송을 보장하는 라우팅 알고리즘을 제안하였고, 알고리즘을 확장하여 패킷 중복 없이 브로트캐스팅을 할 수 있는 기법을 제안하였다[13]. 그러나 이 논문의 결론에서 아직 동적으로 변하는 네트워크까지의 확장은 아직 미해결 문제라고 기술하였다.

Ⅲ. 강건한 신장 트리 알고리즘

1. 제안 기법 개요

본 논문에서는 무선 ad-hoc 망에서 네트워크 위상의 변화에 영향을 적게 받는 강건한 신장 트리를 구성하고, 시간과 메시지 비용 면에서 효율적으로 신장 트리를 유지하는 기법을 제시한다. 무선 ad-hoc 네트워크 환경에서 신장 트리를 구성할 때 위상의 변화로 인해 문제가 되는 시기는 다음 두 가지 경우로 나눌 수 있다.

1) 신장 트리를 구성할 때 네트워크의 위상 변화

2) 신장 트리가 구성된 후 네트워크의 위상 변화

신장 트리를 구성할 때 네트워크의 위상 변화 문제는 네트워크 위상 변경 전의 정보를 이용하여 신장 트리를 구성하고 나서, 변화된 위상 정보에 대해서

는 본 논문에서 제안하는 네트워크 위상의 변경에 따른 알고리즘을 수행하여 최신의 상태를 유지하도록 한다. 물론, 신장 트리를 구성하지 못할 정도로 네트워크 위상의 변경이 발생하면, 그 시점에서 다시 현재 정보를 이용하여 신장 트리를 구성하기 위한 시도를 신장 트리가 구성될 때까지 반복한다. 두 번째인 신장 트리가 구성된 후 네트워크의 위상 변화는 본 논문에서 제안하는 알고리즘을 수행한다. 네트워크의 분할 된 후 다시 신장 트리를 재구성할 때 발생하는 네트워크 위상의 변화 문제는 신장트리를 재구성하는 후 갱신하여 해결할 수 있으나, 각각의 노드들이 유지하고 있는 정보를 이용하여 네트워크 위상의 변경에 따르는 알고리즘 수행함으로서 효율적으로 신장 트리를 재구성하고 유지할 수 있다.

본 논문에서 제안하는 기법에서는 무선 ad-hoc 네트워크 환경에서 강건한 신장 트리를 구성하고 있는 노드들이 유지하고 있어야 할 정보는 다음과 같다.

1) 신장 트리를 처음 구성하는 데 사용되는 변수들
2) 신장 트리를 구성하는 노드들의 집합
3) 신장 트리 경로에 대한 경로 정보 집합
4) 제안 기법 알고리즘에 사용되는 변수들

각 노드들이 우선 지녀야 할 정보는 초기에 신장 트리를 구성하는 데 사용되는 변수들이다. 우선 신장 트리에 참여하는 노드들이 총 개수가 필요하고, 각 노드의 상태 정보와 노드와 직접적으로 연결된 간선(edge)들에 대한 상태 정보들이 있다. 또한, 각 노드는 신장 트리에 구성하는 노드들의 집합을 신장 트리 경로 상의 링크를 기준으로 여러 개의 연결 요소(connected component)들로 나누어 유지한다. 본 논문에서는 다음과 같이 정의하여 사용한다.

CC_j : 노드가 신장 트리 상으로 인접한 j 노드 방향에 있는 신장 트리 연결 요소들의 집합

신장 트리를 구성하는 각 노드는 신장 트리 경로 상에 인접한 노드들에 대한 경로 정보들을 유지하고 있어야 한다. 본 논문에서는 P_{ij}로 표시하며 다음과 같이 정의하여 사용한다.

P_{ij} : 신장 트리 노드 i에서 신장 트리 노드 j까지의 경로 정보 집합

본 논문에서는 P_{ij}를 통해 노드 i와 j의 결합도를 측정한다. 또한, 신장 트리를 구성하는 각 노드들은 본 논문에서 제안하는 알고리즘을 운용하기 위해 사용되는 변수들도 유지해야 한다. 이 외에도 신장 트리를 구성하는 노드가 분산 알고리즘의 효율성을 위해 유지해야 하는 정보는 원래의 신장 트리 경로에 문제가 발생할 경우 원래의 신장 트리 경로를 대체할 수 있는 후보 신장 트리 경로에 대한 경로 정보이다. 본 논문에서는 $Cand(CC_j)$로 표시하며 다음과 같이 정의한다.

$Cand(CC_j)$: 현재의 노드와 신장 트리 경로 상에서 직접 이웃하지 않은 CC_j에 속한 노드들과의 경로 정보 집합 중에서 현재 신장 트리를 대체할 수 있는 후보 경로 집합

CC의 원소의 개수가 2개 이상인 곳에만 Cand(CC)가 필요하며, 각 노드는 최대 CC의 개 수 만큼 후보 신장 트리 경로 정보를 가질 수 있다.

2. 제안 기법 시나리오

본 논문에서 제안하는 기법의 알고리즘을 소개하기 전에 제안 기법에 대한 이해를 쉽게 하기 위해 강건한 신장 트리를 구성하고 유지하는 시나리오를 먼저 설명한다. 설명을 쉽게 하기 위해 전체 네트워크를 신장 트리가 이미 구성된 곳과 신장 트리를 구성하기 위해 분산 알고리즘이 수행되는 곳으로 그림 1과 같이 나누어서 분산 알고리즘이 동작하는 과정을 설명한다.

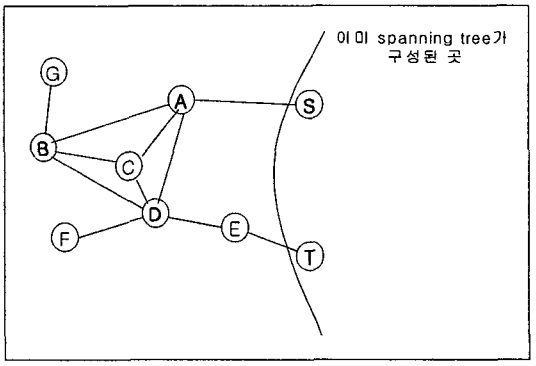


그림 1. 신장 트리를 구성 중인 무선 ad-hoc 네트워크

신장 트리를 구성하는 노드들의 집합은 응용에 따라 ad-hoc 네트워크 노드들 중에서 결정되며, 그림 1에서는 {A, B, D, S, T, ...}이다. 무선 ad-hoc 네트워크가 그림 1과 같고 응용에 따라 신장 트리를 구성하는 노드들의 집합이 {A, B, D, S, T, ...} 일 때 최초로 신장 트리를 구성하여 본 논문에서 제안하는 알고리즘에서 사용되는 정보를 추가하여 그림 2에 나타내었다. 신장 트리 경로는 SA, AB, 와 AD 등이며, 신장트리에 포함되는 각 노드들의 CC와 P_{ij}도 같이 나타내었다.

각 노드들이 만약 알고리즘의 효율성을 위해 Cand 정보를 유지한다면 노드 A는 CC_B나 CC_D의 갯수가 1이기 때문에 필요 없지만, CC_S의 갯수는 1보다 크기 때문에 Cand(CC_S)에 정보를 유지할 수 있다. 노드 B는 Cand(CC_A)로서 P_{BD}를 가질 수 있으며, 노드 D처럼 P_{CA}에 대한 대체 경로로서 P_{DB}나 P_{DT}를 가질 수 있을 경우에는 더 결합도가 높은 경로를 선택하여 Cand(CC_A)에 정보를 유지한다.

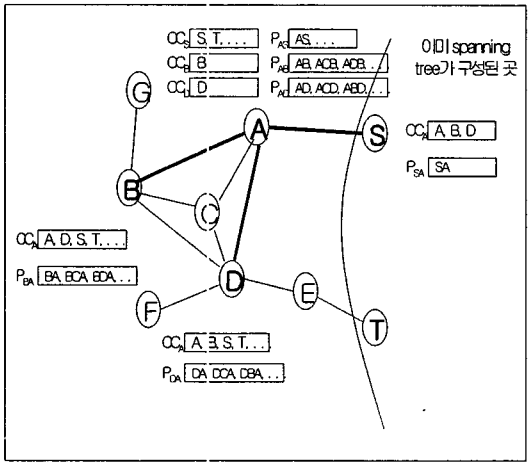


그림 2. 신장 트리를 최초로 구성한 무선 ad-hoc 네트워크

이제 네트워크의 위상이 동적으로 자주 변경되는 무선 ad-hoc 네트워크 환경을 네트워크의 위상이 변화의 정도에 따라 세 종류로 나누어, 각 경우에 대해 신장 트리를 구성하고 유지하는 시나리오를 설명할 것이다. 첫 번째 경우는 우선 신장 트리 경로에 영향을 거의 안주는 간선의 오류나 복구인 경우이며, 두 번째 경우는 신장 트리 경로에 영향을 미치는 간선의 오류나 복구인 경우이며, 마지막으로 세 번째 경우는 신장 트리가 분할되는 경우이다.

신장 트리 경로에 영향을 거의 안 주는 간선의 오류나 복구인 경우, 신장 트리 경로는 그대로 두고 오류나 복구된 간선에 대한 정보를 얻을 수 있는 노드들은 자신의 P_{ij}를 갱신하는 일을 수행한다. 이 과정은 차후에 신장 트리를 구성하고 유지하는 데에 사용될 정보를 보다 정확하게 유지하기 위해서이다. 위의 그림 2에서 간선 CB에 오류가 발생한 경우가 이에 해당한다.

신장 트리 경로에 영향을 미치는 간선의 오류나 복구인 경우는 오류나 복구된 간선의 경로 정보를 갱신하는 일을 수행한 후에 신장 트리 경로를 변경하는 작업이 수행될 것이다. 위의 그림 2에서 간선 AD가 오류가 발생된 경우를 예로 들 수 있다. 이 정보를 알게 된 노드들은 자신의 신장 트리 경로 정보를 갱신하고, 오류가 발생한 신장 트리 경로의 양 끝 노드는 신장 트리 경로를 변경하는 작업을 수행하게 된다. 이 경우 CC의 개수가 적은 쪽에서 주체가 되어 CC의 개수가 많은 쪽으로 결합되는 방식이 용이하다. 노드 D는 Cand(CC_A)가 유효하다면 그 경로로 신장 트리 경로를 설정하도록 함으로써 신장 트리를 재구성 할 수 있을 것이다. 만약 Cand(CC_A)가 유효하지 않다면 CC_A에 속하는 노드 중에 가장 자신과 결합도가 높은 노드를 찾아서 신장 트리의 경로를 설정하도록 한다. 신장 트리 경로가 설정되면 이후 노드가 유지하는 정보를 갱신하는 작업을 수행한다. 알고리즘을 수행한 결과는 그림 3에 도식하였다.

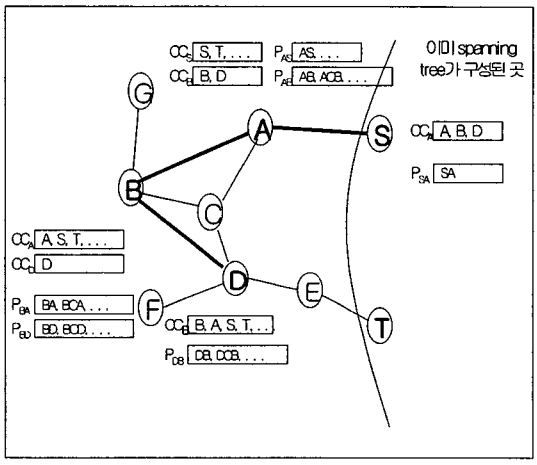


그림 3. 신장 트리 경로에 영향을 주는 오류에 대한 대응 결과

신장 트리가 완전히 절단되는 간선의 오류가 발생한 경우는 오류가 난 간선의 경로 정보를 갱신하는 일을 수행한 후에 신장 트리 경로를 변경하는 작업이 수행될 것이다. 두 번째 경우와 같은 방법으로 해결을 시도한 후에, 성공하지 못하면 오류가 발생한 간선 쪽을 제외한 모든 CC에게 신장 트리의 재구성을 요청한다. 즉, 오류가 발생한 간선 쪽을 제외한 CC에 속한 노드에서 오류가 발생한 간선 쪽 CC에 속한 노드로 연결되는 경로 중에 가장 결합도가 높은 노드를 찾은 후, 그 노드를 통해 신장트리를 재구성한다. 신장 트리가 재구성되면 이에 맞게 노드가 유지하는 정보를 갱신하는 작업을 수행한다. 그림 3에서 간선 SA에 오류가 발생한 경우 제안 기

법에 의해 문제를 해결한 결과를 그림 4에 도식하였다. 우선 A 노드가 CC_A에 속한 노드와 연결이 가능하면 그 노드와 신장 트리를 연결한다. 그러나 노드 A가 CC_A에 속한 아무 노드와도 연결이 불가능하면, 노드 B와 노드 D에게 CC_S에 속한 노드와 연결되는 노드 중에 가장 결합도가 높은 경로를 택해 그 경로로 신장 트리 경로를 변경한다. 그림에서는 가장 결합도가 높은 경로로 P_{DT}가 선정되었다.

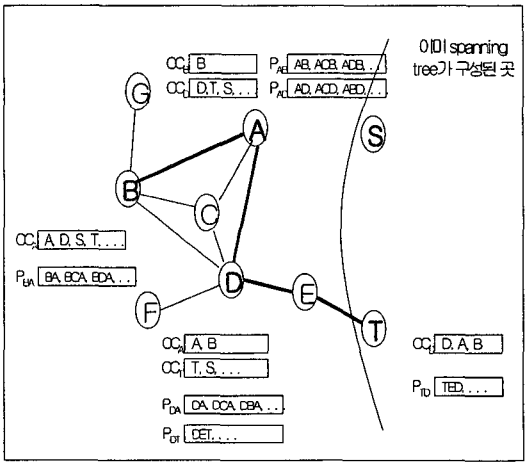


그림 4. 신장 트리가 분할되는 오류에 대한 대응 결과

3. 제안 기법의 분산 알고리즘

본 논문에서 제안한 강건한 신장트리를 유지하기 위해 각 노드가 수행해야 할 알고리즘은 그림 5과 같다. 신장 트리 경로를 대체할 수 있는 후보 신장 트리 경로에 대한 경로 정보인 Cand(CC)는 제안 기법 알고리즘을 단순화하기 위해 고려하지 않았다. 다음은 본 알고리즘에서 사용되는 변수들이다.

E_{ij} : 노드 i와 노드 j를 연결하는 간선

P_{ij} : 노드 i에서 노드 j까지의 신장 트리 경로 정보 집합

$|P_{ij}|$: 노드 i와 노드 j의 결합 정도

CC_j : 노드가 이웃하는 j 노드 방향에 있는 신장 트리 연결 요소들의 집합

CC_j' : j 노드 방향을 제외한 신장 트리의 연결 요소들의 집합

$|CC_j|$: 노드가 이웃하는 j 노드 방향에 있는 신장 트리 연결 요소들의 갯수

ADJ : 임의의 노드에서 신장 트리 상으로 인접한 이웃 노드들의 집합

L_threshold : 결합 정도의 하한 값

$FOUND_{nCCm}$: 노드 n에서 CC_m으로 연결하는 대체 경로를 찾았는지에 대한 부울 변수

STP: 신장 트리 경로의 집합

$U = CC_j + CC_j'$

알고리즘 A.1은 신장 트리를 구성하는 임의의 노드가 자신의 주위에 네트워크 위상이 변경되었음을 감지하였을 때 수행되는 알고리즘이다. 자신이 가지고 있는 신장 트리 경로들에 대해 결합도를 재평가하여 하한 값보다 작은 신장 트리 경로가 있다면 적당한 대체 경로를 찾기 위한 프로토콜을 시작한다. 또한, 아직 대체 경로를 찾지 못해 $FOUND_{nCCm}$ 부울 변수 값이 아직 거짓인 값들에 대하여 자신의 노드에서 CC_m에 있는 노드들과의 결합도를 평가하여 하한 값보다 더 큰 노드가 있다면 대체 경로를 찾은 경우로 간주하여 found_suitable_band_path 메시지를 노드 n에게 보낸다.

알고리즘 A.2는 대체 경로를 찾는 과정이며 band 값이 0일 경우에는 원래 경로가 단절되었음을 의미하므로 CC_m과의 결합도가 하한값보다 작은 경우라도 결합도가 0보다 크면 대체 경로를 찾은 경우로 보고 노드 n에게 found_suitable_band_path 메시지를 보낸다. 대체 경로를 찾을 수 없다면 find_suitable_bond_path 메시지를 이 메시지를 받은 노드를 제외한 신장 트리 이웃 노드들에게 전송한다.

알고리즘 A.3은 found_suitable_bond_path 메시지를 받았을 때 수행하는 알고리즘으로 대체 경로를 찾았을 때 수행하는 알고리즘이다. 이 알고리즘은 대체되는 경로의 소스 노드가 대체하는 경로의 소스 노드에게 reconnect_tree_path 메시지를 보낸다. 또한 대체 경로를 찾은 경우이므로 여러 노드에 있는 FOUND 정보를 참으로 해주기 위해 FOUND_status 메시지를 보낸다. 알고리즘 A.4는 이러한 메시지를 받았을 때 FOUND 정보를 참으로 해주는 알고리즘이다.

알고리즘 A.5는 reconnect_tree_path 메시지를 받았을 때 수행하는 알고리즘으로 신장 트리 경로를 추가하고 노드가 유지하고 있는 관련된 정보를 수정한 후, 관련 정보를 변경할 필요가 있는 노드에게 update_related_information 메시지를 보내는 알고리즘이다. 알고리즘 A.6은 update_related_information 메시지를 받았을 때 관련 정보를 변경하는 알고리즘이다.

Ⅳ. 성능 평가 및 분석

1. 성능 평가 모델

A.1 When node r receives a message that E_{ij} is changed
delete all paths containing edge E_{ij} in P_{nm}, $P_{nm} \in$ STP
while ($|P_{nm}| <$ L_threshold)
if ($|CC_m > | \sum_{i \in ADJ, i \neq m} (CC_i) |$) /* small CC is absorbed to large CC */
sends a find_suitable_bond_path($_nCC_m'$, n, $|P_{nm}|$) message to itself;
while ($FOUND_{sCCm}$ == false) /* for recovery */
if ($\exists k \in CC_m$ such that $|P_{nk}| >$ L_threshold)
sends a found_suitable_bond_path(n, k, $|P_{nk}|$) message to s

A.2 When node s receives a find_suitable_bond_path(CC_m', n, bond) message from node a
$CC_m = U - CC_m'$; $FOUND_{nCCm}$ = false;
t = MAX { $|P_{sc}|$ for $c \in CC_m$ };
if ($t >$ L_threshold)
sends a found_suitable_bond_path(s, c, t) message to n;
else if (bond == 0 and $t > 0$)
sends a found_suitable_bond_path(s, c, t) message to n;
else {
while ($d \in$ (ADJ - {a})) {
sends a find_suitable_bond_path(CC_m', n, $|P_{nm}|$) message to d;
}
}

A.3 When node n receives a found_suitable_bond_path(s, c, t) message
if ($FOUND_{nCCm}$ == false) { /* don't care t, because if $t>0$ ==> topology is changed */
$FOUND_{nCCm}$ = true;
while ($a \in$ ADJ) /* erase all FOUND inf */
sends a FOUND_status(n, CC_m) message to a;
sends a reconnect_tree_path(n, s, CC_m', m, c) message to s;
} else
NOOP;

A.4 When node t receives a FOUND_status(n, CC_m) message from a
$FOUND_{nCCm}$ = true;
while ($d \in$ (ADJ - {a})) {
sends a FOUND_status(n, CC_m) message to d;
}

A.5 When node r receives a reconnect_tree_path(n, s, CC_m', m, c) message
$CC_m = U - CC_n'$
if ($r \neq c$) { /* r == s */
STP = STP + P_{rc}; $CC_c = CC_m$;
while ($a \in$ ADJ)
if ($n \in CC_a$)
$CC_a = CC_a - CC_m$;
sends a reconnect_tree_path(n, s, CC_m', m, c) message to c;
sends a update_related_information(n, CC_m, m) message along tree path from r to n;
} else { /* r == c */
STP = STP + P_{rs}; $CC_s = CC_m'$;
while ($a \in$ ADJ)
if ($n \in CC_a$)
$CC_a = CC_a - CC_m'$;
sends a update_related_information(m, CC_m', n) message along tree path from r to m;
}

A.6 When node t receives a update_related_information(n, CC_m, m) message from a;
If ($t \neq n$) {
$CC_a = CC_a + CC_m$;
while ($d \in$ ADJ)
if ($n \in CC_d$)
$CC_d = CC_d - CC_m$;
sends a update_related_information(n, CC_m, m) message along tree path from t to n;
} else {
STP = STP - P_{tn}; $CC_a = CC_a + CC_m$; $CC_m = \varnothing$;
}

그림 5. 강건한 신장 트리를 유지하는 알고리즘

이 장에서는 본 논문에서 제안하는 강건한 신장 트리 기법을 평가하기 위해서 복잡한 무선 ad-hoc 네트워크를 노드의 이동 모델과 경로 설정 모델로 단순화하여 성능 평가를 실시하였다. 성능 평가의 주요 대상은 본 논문에서 제안하는 강건한 신장 트리 기법에 의해 생성되는 신장 트리가 링크 오류가 발생했을 때 재설정 프로시저를 두어 신장 트리를 재구성하는 기존 기법의 신장 트리보다 네트워크 위상 변화에 덜 영향을 받는 강건한 구조로 구성되는 가를 평가하는 것이다. 부가적인 성능 평가의 대상으로는 제안 기법에 의해 구성된 신장 트리가 특정 응용을 운용하기에 효율적인 구조를 지니고 있는가를 평가하는 것이며, 제안 기법 알고리즘의 동작하면서 낭비되는 메시지 양을 평가하는 것이다.

성능 평가 환경은 무선 노드의 개수는 50개로 설정하고, 각 노드는 직사각형 (1500m × 900m)내의 어느 위치로도 이동이 가능하며, 각 노드가 직접적으로 통신할 수 있는 노드의 거리는 250m 이내로 제한하였다. 신장 트리를 구성하는 노드의 수는 50개의 무선 노드 중에서 10개를 선택하였다. 무선 ad-hoc 네트워크를 단순화하기 위해 기존의 여러 ad-hoc 경로 설정 프로토콜을 고려하지 않고 단순히 BFS(Breadth First Search)를 이용한 경로 설정 모델을 사용하였다. 임의의 소스 노드 A에서 목적지 노드 B까지의 경로를 찾기 위해 BFS를 이용하여 찾으며, 알고리즘의 효율성을 위해 네트워크에 존재하는 노드 A에서 노드 B까지의 모든 경로를 찾는 것이 아니라 BFS로 경로 찾기를 하면서 경로의 길이가 길어지는 경로는 경로 정보로 사용하지 않았다.

시스템 내에 있는 각각의 노드들은 임의 중간기점(random waypoint) 모델에 따라 이동한다. 이동 시나리오는 중단 시간(pause time)에 따라 특성이 다르게 나타난다. 각 노드는 중단 시간 동안 정지하다가 임의 목적지를 직사각형 (1500m × 900m) 공간 내에서 선택한 후에 임의의 속도로 목적지까지 이동한다. 목적지에 도착한 노드는 중단 시간(pause time)만큼 정지해 있다가 새로운 목적지를 선택한 후에 다시 임의의 속도로 이동한다[14].

2. 평가 결과 및 분석

제안 기법에 대한 성능을 평가하기 위하여 이산 사건 시뮬레이션(event-driven simulation) 도구 중에 하나인 SIMLIB을 이용하여 위에서 설명한 이동 모델과 통신 모델과 에너지 모델을 모두 구성하였다[15].

시뮬레이션 매개 변수들은 다음과 같다. 무선 ad-hoc 네트워크의 크기는 1500m × 900m 이며, 각 노드의 전송 범위는 250m로 통일하였다. 전체 시스템에 운용되는 노드의 수는 50개로 초기화하였고, 신장 트리를 구성하는 노드의 수는 10개로 하였다. 노드의 속도는 균등분포로 모델링하였고 노드의 중단 시간은 10초로 하였다. 시뮬레이션 시간은 5000초로 하였다. 경로의 결합도는 경로 정보 집합에 있는 각 경로에 대해 그 경로의 길이를 Lp이라고 할 때 각 경로의 길이에 대해 $1 / 2^{(Lp - 1)}$ 값을 더한 총합으로 하였다. 결합 정도의 하한 값으로는 1을 사용하였다.

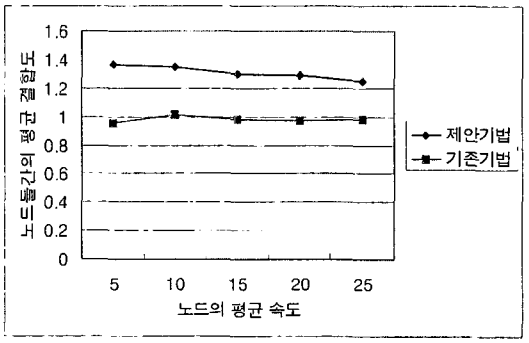


그림 6. 노드들의 이동 속도에 따른 노드들간의 평균 결합도

제안 기법과 기존 기법에 의해 생성된 신장 트리의 강건성을 비교 평가하기 위해 그림 6에 노드의 평균 속도에 따른 제안 기법과 기존 기법의 각 노드들간의 평균 결합도를 나타내었다. 그림에서 알 수 있듯이 제안 기법이 기존 기법에 비해 신장 트리 노드들간의 결합도가 크다는 것을 알 다. 즉, 제안 기법을 사용하면 네트워크 위상 변화에 덜 영향을 받는 강건한 신장 트리를 구성할 수 있음을 알 수 있다. 또한, 기존 기법은 노드의 평균 속도가 증가하더라도 결합도가 일정한 값을 유지하지만, 제안 기법에서는 결합도가 서서히 감소하는 것을 알 수 있다. 이 결과로 알 수 있는 것은 제안 기법은 네트워크의 위상 변화가 빠르게 일어나는 환경에서도 잘 적용할 수 있다는 것을 알 수 있다.

제안 기법에 의해 발생하는 오버헤드를 평가하기 위해 그림 7에 노드의 평균 속도에 따른 제안 기법이 대체 경로를 찾는 알고리즘을 수행한 횟수를 나타내었다. 그림에서 알 수 있듯이 네트워크의 위상 변화가 빠르게 변경되는 환경에서는 제안 기법의 알고리즘의 수행 횟수가 증가하고 있음을 알 수 있

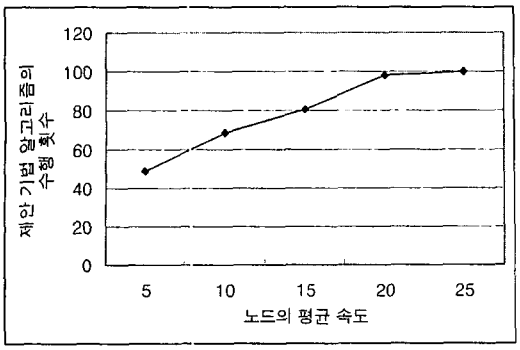


그림 7. 노드들의 이동 속도에 따른 제안 기법 알고리즘 수행 횟수

다. 노드의 평균속도가 20 이상인 경우에는 노드의 평균 속도가 5인 경우보다 두 배 이상 증가했지만 전체 성능 평가 시간이 5000초이므로 제안 기법의 알고리즘의 수행 빈도는 평균적으로 50초에 많아야 한 번 정도 발생하는 것이므로 시스템의 성능에 크게 낭비를 가져오지는 않을 것이다.

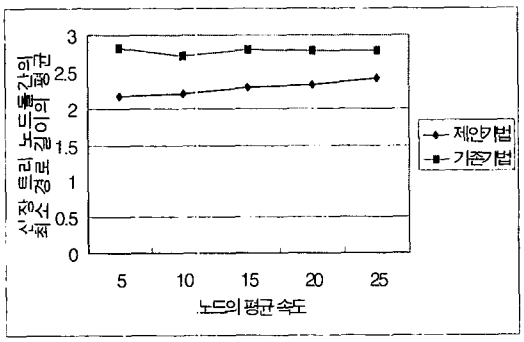


그림 8. 노드들의 이동 속도에 따른 노드들간의 최소 경로 길이의 평균

제안 기법과 기존 기법에 의해 생성된 신장 트리의 효율성을 비교 평가하기 위해 그림 8에 노드의 평균 속도에 따른 신장 트리 각 노드들간의 최소 경로 길이의 평균을 나타내었다. 그림에서 알 수 있듯이 제안 기법이 기존 기법에 비해 신장 트리 노드들간의 경로가 더욱 짧다는 것을 알 수 있다. 즉, 제안 기법에 의해 생성된 신장 트리를 사용하여 분산 응용을 수행하면 더 짧은 경로를 가지고 효율적으로 운용할 수 있다는 것을 알 수 있다.

본 논문에서 제안한 알고리즘을 평가하기 위해 성능 평가를 수행하여 본 논문에서 제안하는 강건한 신장 트리 기법에 의해 생성되는 신장 트리가 기존 기법의 신장 트리보다 네트워크 위상 변화에 덜 영향을 받는 강건한 구조로 구성된다는 것을 보였으며, 제안 기법에 의해 구성된 신장 트리가 특정 응용을 운용하기에 효율적인 구조를 유지하고 있음을 보였다. 또한, 제안 기법에 의해 발생하는 오버헤드는 네트워크 위상이 빨리 변경될수록 커지지만 전체 신장 트리 운용시간에 비해 그렇게 크지 않음을 보였다.

V. 결 론

무선 ad-hoc 네트워크는 어떠한 하부 구조와 중앙 관리의 도움 없이 임시 네트워크를 구성하는 무선 이동 호스트들의 집합이다. 이러한 환경에서 어떤 분산 응용을 수행하기 위해서는 어느 정도 오류를 감내할 수 있는 강건한 구조가 요구되며, 또한 응용을 수행하는 데 사용되는 자원의 효율성도 요구된다. 본 논문에서는 무선 ad-hoc 네트워크 환경에서 강건하고 효율적으로 분산 응용이 수행될 수 있도록 강건한 신장 트리를 구성하고 유지하는 기법을 제안하였다. 성능 평가를 통해 본 논문에서 제안하는 기법이 기존 기법에 비해 더욱 강건하게 신장 트리를 유지하며, 분산 응용을 효율적으로 동작시킬 수 있는 신장 트리를 구성하고 있음을 보였다. 따라서, 무선 ad-hoc 네트워크 환경에서 본 논문에서 제안한 기법으로 신장 트리를 유지한다면 네트워크 위상의 변화에 덜 영향을 받으면서 효율적으로 분산 응용을 수행할 수 있을 것이다.

참 고 문 헌

[1] J. Broch, D. B. Johnson, D. A. Maltz, Yih-Chun hu, and J. Jetcheva", A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", http://www.monarch.cs.cmu.edu/papers. html.

[2] B. R. Badrinath, Arup Acharya and Tomasz Imielinski, "Structuring Distributed Algorithms for Mobile Hosts", *The 14th International Conference on Distributed Computing Systems,* June 1994.

[3] B. M. Maggs and S. A. Plotkin, "Minimum-cost Spanning Tree as a Path Finding problem," *Information Processing Letters,* Vol. 26, January 1988.

[4] R. G. Gallager, P. A. umblet, and P. M. Spira,

"A Distributed Algorithm for Minimum-Weight Spanning Trees", *ACM Transactions on Programming Languages and Systems*, Vol. 5, No. 1, January 1993.

[5] Baruch Awerbuch, Israel Cidon, and Shay Kutten, "Communication-Optimal maintenance of replicated Information," *In Proc. of the 31st IEEE Ann. Symp. on Foundation of Computer Science,* October 1990.

[6] Baruch Awerbuch, Oded Goldreich, David Peleg, and Ronen Vainish", A Tradeoff between information and communication in broadcast protocols," *Journal of ACM,* vol. 37, No. 2, April 1990.

[7] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control", *Communication of the ACM,* Vol. 17, No. 11, November 1974.

[8] M. Schneider, "Self-stabilization", *ACM Computing Surveys,* Vol. 25, No. 1, March 1993.

[9] S. Aggarwal and S. Kutten, "Time optimal self-stabilizing spanning tree algorithms", *13th Conference on Foundations of Software Technology and Theoretical Computer Science,* 1993.

[10] S. Radhakrishnan, Gopal Racherla, C. N. Sekharan, N. S. V. Rao, and Steven G. Batsell, "DST - A Routing Protocol for Ad Hoc networks Using Distributed Spanning Trees", *IEEE Wireless Communications and Networking Conference,* September 1999.

[11] Sandeep K. S. Gupta and Pradip K. Srimani, "Mobility Tolerant Maintenance of Multi-cast Tree in Mobile Multi-hop Radio Networks", *International Conference on Parallel Processing,* September 1999.

[12] U. C. Kozat, G. Kondylis, B. Ryu and M. K. Marina, "Virtual Dynamic Backbone for Mobile Ad Hoc Networks", *IEEE International Conference on Communications,* June 2001.

[13] P. Bose, P. Morin, I. Stojmenović and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks", *Wireless Networks* Vol. 7, November 2001.

[14] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks", in *Mobile Computing,* T. Imielinski and H. Korth, editors, Kluwer, 1996.

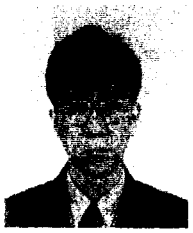
[15] A. Law and W. Kelton, *Simulation Modeling and Analysis,* 2-ed., McGraw-Hill, 1991.

엄 영 익(Young Ik Eom) 정회원

1983년 2월 : 서울대학교 계산통계학과 학사
1985년 2월 : 서울대학교 대학원 전산과학전공 석사
1991년 8월 : 서울대학교 대학원 전산과학전공 박사

현재 : 성균관대학교 정보통신공학부 교수
<주관심 분야> 분산 시스템, 이동 컴퓨팅 시스템, 분산 객체 시스템

강 용 혁(Kang Yong-Hyeog) 정회원



1996년 2월 : 성균관대학교 정보공학과 학사
1998년 2월 : 성균관대학교 정보공학과 컴퓨터공학 전공 석사
2000년 2월 : 성균관대학교 전기전자 및 컴퓨터공학부 박사과정 수료
<주관심 분야> 분산 시스템, 이동 컴퓨팅 시스템, Mobile Ad-hoc networks