블루투스 장치 간의 평균 홉 수를 줄이기 위한 트리 스캐터넷 형성 알고리즘

강 승 호[†] · 강 대 욱[††] · 임 형 석[†††]

요 약

트리 구조를 기반으로 하는 블루투스 스캐터넷은 적은 링크 수와 간단한 라우팅 방법 등 여러 가지 장점을 가지고 있다. 본 논문은 블루투스 규격상의 제약조건을 만족시키면서 노드 간 평균거리가 가장 짧은 트리형태에 가까운 스캐터넷을 형성하는 알고리즘을 제시한다. 그리고 제시한 방법이 기존의 방법보다 지연시간을 더 초래하지 않고도 장치 간 평균 홉 수를 감소시킬 수 있음을 시뮬레이션을 통하여 보인다.

A Tree Scatternet Formation Algorithm for Reducing Average Hop Count Between Bluetooth Devices

Seung-Ho Kang[†] · Dae-Wook Kang[††] · Hyeong-Seok Lim[†††]

ABSTRACT

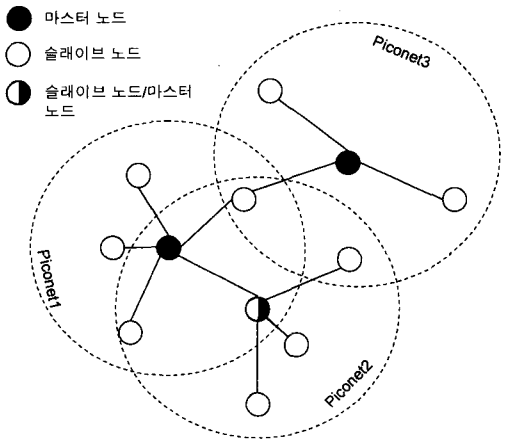
The Bluetooth Scatternet based on tree structure has several merits such as small number of links and simple routing method. This paper proposes an algorithm which satisfies the constraints of Bluetooth Specification and forms the topology of scatternet as close as the tree structure that has the shortest average inter-node distance among trees. Also we show that the proposed method reduces the average hop count between any two devices compared to early works without higher formation delay by a simulation.

키워드 : 블루투스(Bluetooth), 스캐터넷 형성 알고리즘(Scatternet Formation Algorithm), Ad Hoc 네트워크(Ad-Hoc Network), 위상(Topology)

1. 서 론

블루투스는 2.4GHz ISM 주파수대에서 주파수 홉핑 기술을 채택하고 기존의 Ad Hoc 네트워크 기술과는 달리 마스터-슬레이브 매커니즘에 기반한 TDD(Time Division Duplex) MAC 프로토콜을 사용한다[2, 3]. 홉핑 기술의 특성상 의사 랜덤한 주파수의 시퀀스는 도청하기 어렵고 같은 주파수대를 사용하는 이 기종 또는 동 기종의 장치들로부터의 간섭에 강하다[3, 5]. 따라서 일정한 영역 안에 다수의 블루투스 장치 간의 통신이 가능한데 이에 대한 효율적인 통신 방법이 필요하게 되었다.

블루투스의 네트워크는 (그림 1)처럼 한 개의 마스터와 최대 7개의 슬레이브로 구성되는 피코넷(piconet)과 두 개 이상의 피코넷들로 구성된 스캐터넷(scatternet)으로 구분된다. 블루투스 네트워크에서의 효율적인 통신 방법에 대해서는 크게 두 가지로 연구가 진행되고 있다[13]. 그 중 하나는 피코넷 내부에서의 링크 스케줄링에 관한 것이고 다른 하나는 스캐터넷의 형성 알고리즘과 라우팅에 관한 것이다. 현재 여러 스캐터넷 형성 알고리즘들이 제시되어 있다[7, 9, 11, 15, 16].



(그림 1) 세개의 피코넷으로 구성된 스캐터넷

※ 이 논문은 2001년도 전남대학교 학술연구비 지원에 의하여 연구되었음.
† 준 회 원 : 전남대학교 대학원 전산학과
†† 종신회원 : 전남대학교 컴퓨터정보학부 교수
††† 정 회 원 : 전남대학교 전자컴퓨터정보통신공학부 교수
논문접수 : 2003년 10월 28일, 심사완료 : 2004년 4월 26일

스캐터넷 형성 방법 중에는 스캐터넷내의 링크 수 최소화, 피코넷을 연결하는 브리지 노드의 참여 피코넷 수 최소화, 그리고 노드 간 경로의 유일성 등의 특성에 근거한 트리 형태가 주목을 받고 있다[10, 11, 16]. 링크 수가 작으면 노드 간 링크의 연결과 스케줄링에 소모되는 시간이 그만큼 작다. 또한 브리지 노드는 자신이 속한 여러 피코넷들 사이에서 시간 분배를 통해 각 피코넷에 참여하게 되므로 참여하는 피코넷 수가 적을수록 스캐터넷 전체의 정보 전송의 효율성이 나아진다. 그리고 각 노드 간 경로가 유일하면 라우팅 알고리즘이 단순하다.

트리 기반 스캐터넷 형성 알고리즘들[10, 11, 16]은 완전한 연결성을 보장하고 확장성을 가지며 새로운 장치들의 추가, 삭제 등에 의해 발생하는 토폴로지의 동적인 변화에도 적절한 대처 방법을 제시하고 있다. 그러나 이들은 네트워크의 중요한 성능 평가 요소 중의 하나인 장치 간의 홉 수를 적극적으로 고려하고 있지 않다. 장치들이 많고 교환되는 데이터양이 많다면 스캐터넷의 형태에 의해 좌우되는 홉 수는 전체적인 네트워크의 효율성에 큰 영향을 미치게 된다. 본 논문은 장치간의 평균 홉 수를 적극적으로 반영하는 트리 형태의 스캐터넷 형성 알고리즘을 제시한다. 이 알고리즘은 임의의 두 노드 간 평균 거리를 최소로 하는 트리 형태를 고려하여 스캐터넷을 이 형태에 가깝게 형성함으로써 장치 간의 평균 홉 수를 줄일 수 있다. 또한 형성 상에서의 추가적인 지연시간을 초래하지 않는다.

본 논문의 구성은 다음과 같다. 우선 2장에서 지금까지 제시된 여러 연구 결과 중 본 논문과 직접적으로 관련이 있는 스캐터넷 형성 알고리즘들을 검토하고 3장에서는 본 논문이 제시하는 알고리즘을 기술한다. 4장에서는 *Network Simulator*(ns-2)[8]로 작성한 시뮬레이터를 이용하여 기존의 알고리즘과 성능을 비교, 분석하고 5장에서는 결론과 향후 연구 과제를 제시한다.

2. 관련 연구

스캐터넷 형성 알고리즘은 사전 설정 없이 Ad Hoc한 방법으로 완전한 연결성을 보장하고 형성 과정상에서 적은 지연시간만을 소비하며 연결 이후 데이터 전송 시에도 보다 나은 효율성을 갖아야 한다. 제시되어있는 프로토콜들로는 Bluetree, BTCP, TSF등이 있다.

1.1 Bluetree

Bluetree[16]는 일정한 영역내의 블루투스 장치들 간에 Inquiry 단계가 끝나서 자신과 한 홉 거리에 있는 장치들의 주소와 클럭를 안다는 상태에서 시작한다. 우선 임의의 노드를 blueroot로 삼아 이 노드가 자신과 한 홉 거리에 있는 노드들을 대상으로 피코넷을 형성하도록 한다.

최초로 형성된 피코넷내에서 슬레이브 역할을 하는 노드들은 아직 네트워크에 연결되지 못한 상태이면서 자신과 한 홉 거리에 있는 노드들을 대상으로 다시 연결을 시도한다. 이런 식으로 일정한 영역내의 모든 노드들이 연결될 때까지 진행 되는데 이는 그래프 이론에서 신장 트리를 구하는 방식과 비슷하다.

이 프로토콜은 앞에서 언급한 트리 토폴로지가 갖는 장점들을 가지고 있다. 그러나 데이터 통신의 효율성 측면에서 최종적으로 형성된 네트워크의 위상이 갖는 중요성 못지않게 형성 과정상에서 소요되는 시간 또한 중요한 의미를 갖는다. 특히 블루투스는 장치 간 연결이 이루어지는데 드는 지연시간의 대부분이 Inquiry 단계에서 발생하는데도 이에 대한 고려 없이 Inquiry 단계를 사전에 가정 해버렸다는 문제점이 있고, 또한 어떤 노드가 blueroot의 지위를 갖게 되는지에 대한 명확한 기준이 없다. 그리고 Inquiry나 Page 단계에서 노드가 수행할 역할이 사전에 임의로 고정되어 있어서 Ad Hoc한 상황을 적절히 반영하지 못하고 있다.

1.2 BTCP(Bluetooth Topology Construction Protocol)

BTCP[9]는 각 노드의 역할이 고정되어 있음을 가정하지 않고 랜덤한 시간 간격으로 INQUIRY 상태와 INQUIRY SCAN 상태를 번갈아 수행한다고 가정한다. 그리고 스캐터넷을 형성하는데 다음과 같은 세 가지 단계를 거친다.

단계 I : 조정자 선출 - 일정한 영역내의 모든 노드는 초기값이 1인 VOTES라는 변수를 가지며 다른 노드와 연결을 수행한 후 이 값을 비교한다. 이때 VOTES 변수 값이 큰 쪽은 승자가 되어 패자가 보유하고 있는 정보를 가져오고 또 다른 노드와 대결을 시도한다(VOTES 변수 값이 같다면 주소 값이 큰 쪽이 승자가 된다). 만약 노드 수가 n개이면 n-1번의 비교를 통해 최종적인 승자가 선출되고 이 노드가 장차 스캐터넷 형성을 주도할 조정자가 된다.

단계 II : 역할 결정 - 단계 I을 거쳐 선출된 조정자는 전체 노드의 수와 각 노드의 주소, 클럭 정보를 가지게 된다. 이 조정자는 이렇게 모아진 정보를 바탕으로 영역내의 최소수의 피코넷 수 P를 아래의 식에 의해 구해낸다.

$$P = \left| \frac{17 - \sqrt{289 - 8N}}{2} \right|, \quad 1 \le N \le 36$$

조정자는 자기 자신과 P-1개의 잠정적인 마스터와 P(P-1)/2개의 브리지 노드를 지정한다. 그리고 잠정적인 마스터들에게 일시적인 연결을 하고 SLAVELIST와 BRIDGELIST를 전달한 다음 연결을 끊는다.

단계 Ⅲ : 실제 연결 성립 - 단계 Ⅱ를 거쳐 지정된 잠정적인 마스터들과 조정자는 SLAVELIST와 BRIDGELIST에 있는 노드들을 대상으로 연결을 시도하고 역할을 부여한 다음 최종적인 스캐터넷을 형성한다.

이 프로토콜은 노드의 지위가 사전에 고정되어 있지 않고 INQUIRY 상태와 INQUIRY SCAN 상태에 번갈아 있게 된다는 보다 현실성 있는 가정을 채택하였고 이 프로토콜에 의해서 생성된 스캐터넷이 ① 피코넷을 연결하는 브리지 노드의 참여 피코넷 수 최소화 ② 주어진 노드 수 n개 내에서 최소수의 피코넷을 생성한다는 등의 장점을 가지고 있다.

그러나 이 프로토콜은 전체 노드 수와 관련 정보를 수집하기위해 사전 연결이 있어야 하는데 이는 노드 수가 크면 스캐터넷을 형성하는데 많은 시간을 소요한다. 또한 위 식에서 알 수 있듯이 노드수가 36개 이하인 경우만 가능하고 이를 초과한 경우에는 적절한 방법을 제시하지 못하고 있다[9].

1.3 TSF(Tree Scatternet Formation)

TSF[9, 10]은 BTCP가 제시한 노드의 INQUIRY 상태와 INQUIRY SCAN 상태의 전이 가정을 전제로 하고 있다. 여기에 각 노드를 트리의 형성 과정에서의 역할에 따라 Root 노드, Non-root 노드, 그리고 Free 노드로 나눈다. 일단 장치가 일정한 장소에 등장하게 되면 Free 노드의 지위에서 트리 형성을 시도하게 되고 노드 간 회합이 있게 되면 이때 INQUIRY를 수행하는 쪽은 Root 노드의 지위를, INQUIRY SCAN을 수행하는 쪽은 Non-root 노드의 지위를 갖고 각각의 역할을 수행하게 된다.

그리고 트리 형태의 스캐터넷을 형성하기 위해 다음과 같은 세 가지 규칙에 따라 형성이 진행된다.

① Free 노드는 다른 Free 노드 또는 Non-root 노드하고만 연결을 시도한다.
② Root 노드는 오직 다른 Root 노드하고만 연결을 시도한다.
③ Non-root 노드는 오직 Free 노드하고만 연결을 시도한다.

Non-root 노드는 INQUIRY 상태에만 머물고 Free 노드와 Root 노드는 random(E[T_{inq}], D)의 간격으로 INQUIRY 상태와 INQUIRY SCAN 상태를 번갈아 가며 트리 형성상의 역할을 수행한다.

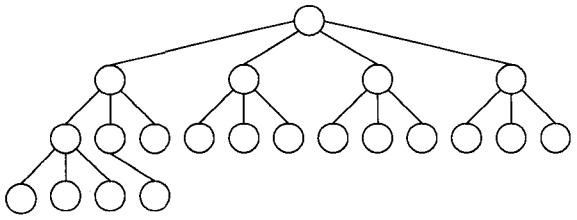
E[Tinq]는 Inquiry 단계를 수행하는데 걸릴 것으로 예상되는 시간으로 시뮬레이션 결과 1.8s 정도이며 D는 임의 간격의 크기로 $2.2 \le D \le 4.4$라고 알려져 있다[11]. 따라서 실제 한 노드가 INQUIRY 또는 INQUIRY SCAN 상태에 머무는 시간은 $2.2 \times E[Tinq] \le random(E[Tinq], D) \le 4.4 \times E[Tinq]$이다.

이 방법은 BTCP가 제시한 역할 전이라는 현실성 있는 가정을 채택하고 노드 수에 관계없이 트리 형태의 스캐터넷 형성이 가능하다는 장점을 가지고 있다. 또한 다른 프로토콜과는 달리 Inquiry 단계를 스캐터넷 형성에 적극적으로 고려하였다. 그러나 이 방법은 Inquiry 단계에서 각 노드가 임의적으로 주어진 시간만을 INQUIRY나 INQUIRY SCAN 상태에서 소모한다고 가정함으로써 스캐터넷의 형태가 단지 트리임을 보장할 뿐 트리 형태 자체도 네트워크의 전체 데이터 전송상의 효율성에 중요한 영향을 미친다는 사실을 적절히 고려하지 못하고 있다.

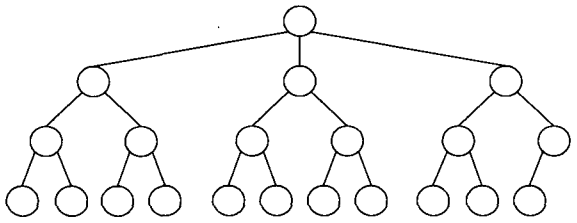
3. 제안 알고리즘

3.1 이론적 배경 - 트리 위상의 특성

n개의 노드와 분지수가 $d(d \ge 3)$ 이하인 트리들 중 노드 간 평균거리가 가장 짧은 형태는 각 노드의 분지수가 d이거나 1이며(마지막 단말 노드의 부모 노드는 예외일 수 있다.) (그림 2)와 같은 형태이다[4]. 예를 들어, 21개의 노드를 가지는 트리 중 분지수가 4 이하인 경우와 3 이하인 경우에 노드 간 평균거리가 최소인 형태가 (그림 2)에 나타나 있다. 이중 (그림 2)(a)가 (그림 2)(b)보다 노드 간 평균거리가 짧다.



(a) 분지수가 4인 트리



(b) 분지수가 3인 트리

(그림 2) 노드 간 평균 거리가 가장 짧은 분지 수 3과 4인 트리

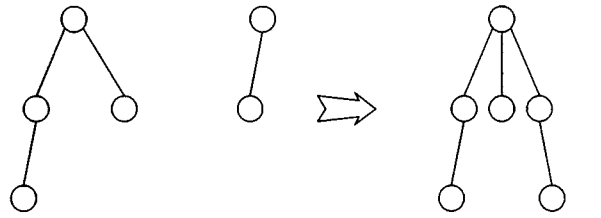
여러 개의 트리로 구성된 포레스트를 단일 트리로 구성해 나갈 때 분지수가 큰 (그림 2)형태의 트리에 가깝도록 구성하면 노드 간 평균 거리가 상대적으로 짧은 단일 트리를 얻을 수 있다. TSF처럼 동적으로 트리 형태의 스캐터넷을 형성하는 것은 여러 개의 트리로 구성된 포레스트를 단

일 트리로 형성하는 것과 유사하다. 따라서 스캐터넷을 분지수가 큰 (그림 2) 형태에 가깝게 형성하면 상대적으로 장치 간 평균 홉 수가 작은 네트워크를 얻을 수 있다.

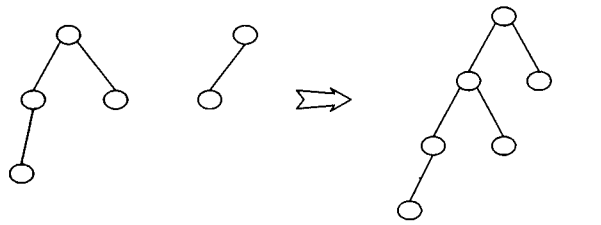
이러한 특성을 반영한 단일 트리 형성 방법으로 트리를 이용한 집합 표현 방식 중 가중법칙[6]을 통해 합집합을 구해내는 방식을 변형한 방법을 제시한다.

방법 1 : 높이가 다른 두 트리의 경우엔 높이가 높은 트리의 루트가 낮은 쪽 트리의 루트를 자식으로 삼는다(그림 3)(a).

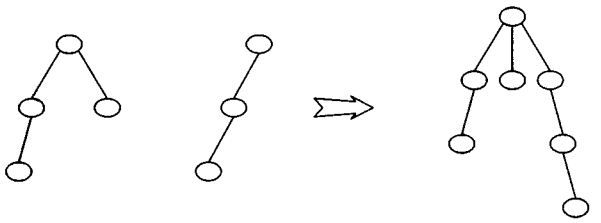
방법 2 : 높이가 같은 두 트리의 경우엔 가중법칙처럼 노드 수가 많은 트리의 루트가 작은 트리의 루트를 자식으로 삼는다(그림 3)(c).



(a)'



(b)



(c)

(그림 3) 포레스트를 단일 트리로 구성하는 방법

트리의 높이가 다른 경우 (그림 3)(b)처럼 낮은 트리가 높은 트리를 자식으로 삼게 되면 형성된 트리의 전체적인 높이가 결합되기 이전의 높은 트리의 높이보다 1만큼 높아지게 된다. 그러나 방법 1처럼 높은 트리의 루트를 결합 이후에도 루트로 삼게 되면 트리의 높이는 결합되기 이전의 높은 트리보다 높아지지는 않는다. 그리고 방법 2처럼 높이가 같은 경우 노드수가 많은 트리가 작은 트리를 자식으로 삼게 하면 트리의 높이는 1만큼 높아지나 분지수가 더 커진다. 방법 1과 방법 2를 이용하면 상대적으로 높이가 낮고 분지수가 큰 단일 트리를 얻을 수 있다. 이는 (그림 2)(a)와 (그림 2)(b) 중 분지수가 큰 (그림 2)(a)형태일 가능성이 높아진다. 따라서 방법 1, 방법 2를 이용해 포레스트를 단일 트리형태로 구성해 나간다면 노드 간 평균 거리가 보다 짧은 형태의 트리를 얻을 수 있다.

3.2 스캐터넷 형성 알고리즘

스캐터넷의 형성 과정은 포레스트를 하나의 트리로 형성해가는 과정과 유사하므로 Root노드 간에 결합이 이루어질 때 방법 1, 방법 2를 이용하면 장치 간 평균 홉 수면에서 보다 개선된 트리 형태의 스캐터넷을 형성할 수 있다. 본 논문이 제시하는 스캐터넷 형성 알고리즘의 기본적인 목적은 TSF의 일반적인 가정과 규칙을 수용하면서도 트리의 이러한 특성에 착안해 노드 간 평균 홉 수를 줄임으로써 전체적으로 데이터 효율성이 나은 네트워크를 형성하려는 것이다.

그러나 앞 절에서 제시한 트리형성 방법을 Inquiry 단계에서 스캐터넷을 형성하는 데에는 그대로 적용할 수 없다. 현재 블루투스 명세서[2]가 제시하는 IAC(Inquiry Access Code)패킷이나 FHS(Frequency Hop Synchronization) 패킷에는 장치 자신이 속한 트리의 높이나 노드 수 등의 정보를 담아 보낼 공간이 할당되어 있지 않기 때문이다. 따라서 스캐터넷 형성 과정에 있는 두 트리의 Root 노드가 연결을 시도할 때 트리의 높이 정보를 비교하는 교섭과정을 밟을 수는 없다.

이러한 명세서상의 제약조건을 만족시키면서도 위에서 제시한 효율적인 트리를 구성하는 방법을 적절히 반영할 수 있는 알고리즘은 다음과 같다.

Free 노드의 지위로부터 INQUIRY를 수행하여 최초로 Root 노드가 된 때부터 각 Root 노드는 자신이 INQUIRY를 수행하는 중 얻어진 FHS 패킷 수를 고려하여 그 패킷 수에 비례하는 일정 시간 Tinq(식 (1))동안 INQUIRY를 수행하도록 하고 패킷 수에 반비례하는 일정 시간 Tinq-scan (식 (2))는 INQUIRY SCAN을 수행하도록 한다. 이 방법은 FHS 패킷 수가 많은 쪽의 Root 노드가 적은 쪽의 Root 노드를 스캐터넷 형성 시에 자식 노드로 삼을 가능성을 크게 한다.

$$Tinq = \alpha \times N_{FHS} \times E(Tinq) \quad (1)$$

$$Tinq\text{-}scan = \beta \times (\gamma - N_{FHS}) \times E(Tinq) \quad (2)$$

여기서 N_{FHS}는 INQUIRY를 수행하면서 모은 FHS 패킷의 수를 나타내고 E(Tinq)는 TSF에서 언급한데로 INQUIRY 단계를 수행하는데 걸릴 것으로 예상되는 시간을 나타낸다. 그리고 α, β는 N_{FHS} 값에 의해 INQUIRY나 INQUIRY SCAN 수행 시간을 구하는데 있어서 이의 영향을 조절할 상수의 파라미터 값이다. γ는 Free 노드가 최초로 Root 노

드가 된 이후 INQUIRY SCAN을 수행할 때 걸리게 될 E(Tinq)의 배수를 나타내며 이후 INQUIRY를 수행하며 N_{FHS}가 증가하게 되면 $\gamma - N_{FHS}$가 작아져서 상대적으로 INQUIRY SCAN을 수행하는데 소비하는 시간은 적어진다. 그리고 Free 노드는 TSF에서 제시한 것처럼 random (E(Tinq), D) 간격으로 INQUIRY와 INQUIRY SCAN을 번갈아 수행한다. Non-root 노드는 Free 노드를 대상으로 INQUIRY만 계속해서 수행한다.

각 노드의 상태-기계를 (그림 4)의 의사코드로 제시한다.

현재의 명세서상 조건을 고려해 볼 때 합병할 두 트리의 Root 노드들은 자신과 한 홉 차이가 있는 노드들의 숫자만을 알 수 있다. 즉 자신들이 이제 까지 INQUIRY를 수행해 오면서 자신의 자식 노드로 삼은 노드들의 개수라는 정보만을 스캐터넷 형성 시에 활용할 수 있다. 이 자식 노드 수가 그 트리의 높이나 전체 노드 수를 정확히 대신 할 수는 없다. 따라서 제시한 단일 트리 형성 방법이 본 논문의 알고리즘에 그대로 적용되었다고 할 수는 없다. 그러나 Root 노드는 Root 노드하고만 연결을 시도할 수 있다는 규칙을 고려해 보면, 현재의 Root 노드의 자식 노드 수가 많다는 사실은 많은 다른 트리들을 자식으로 삼고 있음을 의미한다. 따라서 이런 트리는 자식 노드 수가 적은 Root를 가진 트리보다 상대적으로 트리의 높이가 높고 노드 수가 많을 확률이 그만큼 크다.

또한 INQUIRY 단계에서는 Root 노드들이 자신의 자식 노드들의 수자를 비교해 가며 연결을 시도할 수도 없다. 앞에서도 언급했듯이 사전 교섭이 불가능하기 때문이다. 따라서 다른 전략이 필요한데, 자식 노드가 많은 Root 노드의 INQUIRY 수행 시간을 상대적으로 늘려주고 INQUIRY SCAN 수행 시간은 상대적으로 줄여줌으로써 스캐터넷 형성 시 높이가 높고 노드 수가 많은 트리의 Root 노드가 높이가 낮고 노드 수가 적은 트리의 Root 노드를 자식으로 삼을 가능성을 높여 주는 방법이다. 이렇게 함으로써 TSF처럼 Root 노드가 단지 INQUIRY나 INQUIRY SCAN을 랜덤한 시간만을 수행한 결과로 얻어지는 트리보다 분지수가 큰 (그림 2) 형태의 트리를 얻을 가능성이 높아진다. 이는 곧 스캐터넷 내의 임의의 두 장치 간 홉 수를 감소시킴으로써 보다 효율적인 데이터 통신이 가능하게 됨을 의미한다. 제시한 알고리즘이 실제 트리의 높이를 낮추고 (그림 5)의 트리 형태에 가까워지는 지는 시뮬레이션을 통하여 검증하였다.

```
PROCEDURE FREE() {
    do {
        state <-- OPPOSITE( state )
        t_state <-- random( E[Tinq], D )
        현재 상태에서 t_state 동안 머무르며 state에 따라
        INQUIRY나 INQUIRY SCAN 수행
    } while( !FHS() )
    if ( FHS() == 1 ) {
        지위를 Non-root 로 바꾸고 NON_ROOT() 수행
    }
    else if ( FHS() == 2 ) {
        지위를 Root 로 바꾸고 ROOT() 수행
    }
}

PROCEDURE ROOT() {
    A : do {
        state <-- OPPOSITE( state )
        if ( state == INQUIRY )
            t_state <-- α * N_FHS * E( Tinq )
        else if ( state == INQUIRY SCAN )
            t_state <-- β * ( γ - N_FHS ) * E( Tinq )
        t_state 동안 각 상태에서 state에 따라 INQUIRY나
            INQUIRY SCAN 수행
    } while ( !FHS() )
    if ( state == INQUIRY )
        goto A
    else
        Not-root 노드로 지위 변화한 후
        NON_ROOT() 수행
    }

PROCEDURE NON_ROOT() {
    do {
        state <-- INQUIRY
        INQUIRY 수행
    } while( 1 )
}

PROCEDURE FHS() {
    if ( send FHS packet )
        return 1
    else if ( receive FHS packet )
        return 2
    else
        return 0
}

PROCEDURE OPPOSITE( state ) {
    if ( state == INQUIRY )
        state <-- INQUIRY SCAN
    else
        state <-- INQUIRY
        return state
}
```

(그림 4) 제안 알고리즘에 의한 각 노드의 의사 코드

4. 시뮬레이션 및 성능 비교, 분석

4.1 시뮬레이션 환경

본 논문에서 제시한 알고리즘과 TSF는 IBM에서 제공한 BlueHoc, Bluescat 모듈[1] 부분을 수정하여 구현하였으며 *Network Simulator*(ns-2)[8]를 사용하여 시뮬레이션 하였다.

모든 장치는 TSF의 시뮬레이션에서와 같이 20×20 m의 장소에 랜덤하게 위치하며 15초 이내에 랜덤한 간격으로 진입하는 것으로 하였다[11]. 진입 시 각 장치는 Free 노드의

지위를 갖고 그 초기 상태는 INQUIRY와 INQUIRY SCAN 상태가 각각 50%가 되도록 구성하였다. 그리고 TSF와의 공정한 비교를 위하여 E(Tinq)나 D값은 TSF의 것을 그대로 사용하였다.

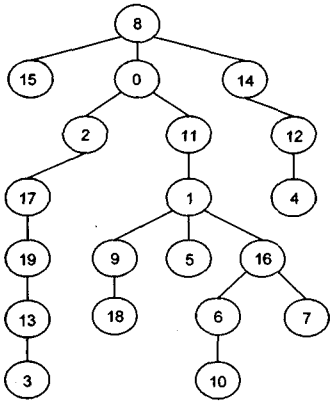
식 (2), 식 (3)의 α, β는 각각 1로 γ는 한 피코넷 내의 최대 장치수인 8로 하였다. α와 β는 임의로 채택한 것이지만 보다 많은 시뮬레이션을 통해 전체 스캐터넷을 형성하는데 걸리는 시간이나 트리의 높이를 낮출 수 있는 적정한 값을 구할 수 있을 것으로 본다.

시뮬레이션을 통한 비교는 INQUIRY SCAN 간격 2.56s마다 실제 scan을 수행하는 기간인 scan window 값을 명세서가 권장하는 18슬롯(11.25ms)과 TSF가 INQUIRY에 소요될 것으로 예측한 E(Tinq)값, 즉 1.8s 두 가지의 경우로 나눠서 각각 실행 하였다. scan window 값은 INQUIRY SCAN 간격과 동일한 시간을 갖도록 할 수도 있지만 전력 소모의 측면이나 이미 네트워크에 연결된 상태의 노드라면 수행해야 할 다른 일들이 있을 수 있기 때문이다. 그리고 기타 INQUIRY 단계에서 사용되는 여러 변수나 파라미터 값들은 명세서상의 기준을 그대로 따르도록 하였다.

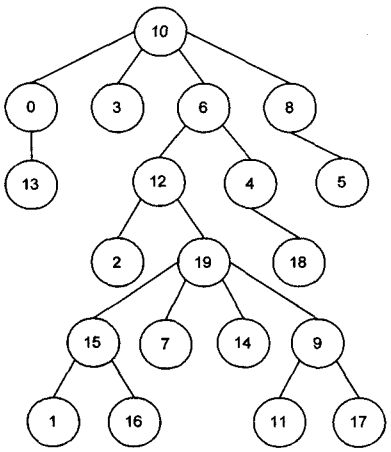
시뮬레이션은 노드 수를 10개에서부터 80개까지 매 10개씩 늘려가며 각각의 방법으로 수행하고 지연시간과 노드 간 평균 홉 수를 측정하였다. 그리고 시뮬레이션의 결과인 그래프의 각 점들은 각각의 방법에 따라 20번씩 수행한 후 그것들을 평균한 것이다.

4.2 TSF와의 성능 비교, 분석

우선 각 알고리즘에 의해 얻어진 스캐터넷의 한 예가 (그림 5)와 (그림 6)에 나타나있다. (그림 5)는 TSF에 의해 얻어진 20개 노드로 구성된 스캐터넷의 한 예이고 (그림 6)은 본 논문이 제안한 알고리즘에 의해 얻어진 스캐터넷의 한 예이다. 그림을 보면 본 논문이 제시한 알고리즘에 의해 구해진 트리 형태의 스캐터넷이 분지수가 큰 (그림 2)의 트리 형태에 보다 가까움을 알 수 있다. 따라서 임의의 두 장치 간 평균 홉 수가 더 적다는 것도 알 수 있다.



(그림 5) TSF에 의해 구해진 노드 수 20개의 스캐터넷 예

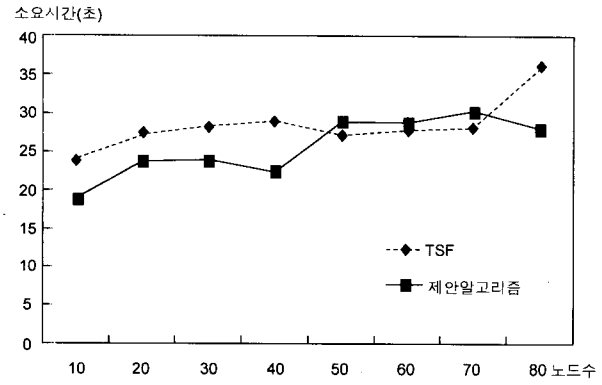


(그림 6) 제안 알고리즘에 의해 구해진 노드 수 20개의 스캐터넷 예

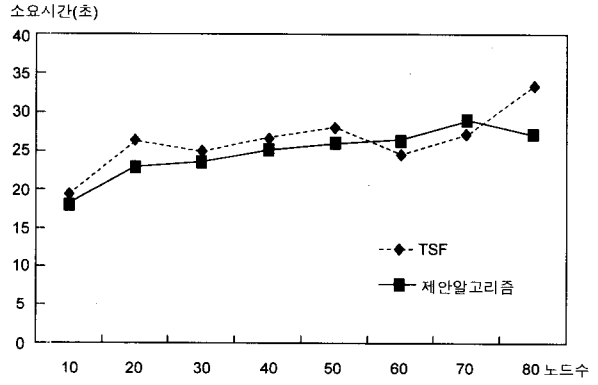
두 장치 간에 연결이 이루어지기까지의 전체 지연 시간은 INQUIRY와 PAGE 단계가 모두 끝날 때까지의 시간이다. 그러나 블루투스 장치 사이에 연결이 이루어 질 때 지연 되는 시간의 대부분은 INQUIRY 단계가 차지한다[3, 5]. PAGE 단계에서는 잠정적인 마스터가 연결하고자 하는 대상, 즉 잠정적인 슬래이브의 주소와 내부클럭을 INQUIRY 단계에서 얻어놨기 때문에 이로부터 슬래이브의 현재 scan 주파수를 바로 알아 낼 수 있다. 따라서 두 블루투스 장치 간의 지연시간을 INQUIRY 단계가 시작된 후 잠정적인 마스터가 잠정적인 슬래이브로부터 FHS 패킷을 받는 순간까지로 제한해도 무리가 없다. 이러한 근거로 본 논문에서 행한 시뮬레이션도 이러한 INQUIRY 단계가 끝나는 시점까지를 지연시간으로 정의하고 측정, 비교하였다.

(그림 7)과 (그림 8)은 scan window가 각각 11.25ms일 때와 1.8s일 때의 스캐터넷이 형성되기까지의 지연 시간을 나타내는 그래프이다. 전체적인 스캐터넷이 형성되기까지에 소요되는 시간은 노드 수가 증가함에 따라 어느 정도 증가하기는 하나 뚜렷하게 영향을 받지는 않는 것으로 나타났다. 이러한 이유는 장치들의 진입 시간이 15초로 제한되어 있고 각 장치들이 병렬적으로 연결을 시도하기 때문이다. 그리고 TSF나 본 논문이 제시하는 알고리즘에 의한 것이나 소요 시간 면에서의 차이는 큰 의미가 없는 것으로 나타났다.

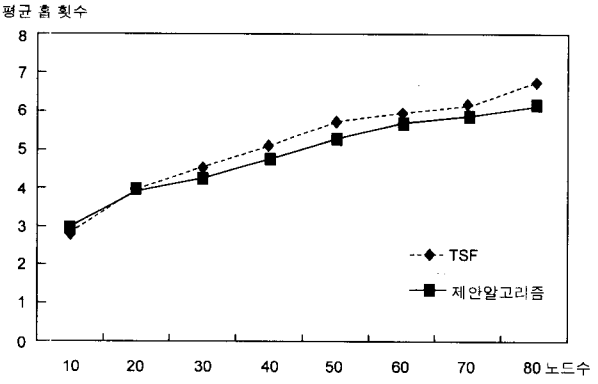
(그림 9)와 (그림 10)은 scan window가 각각 11.25ms와 1.8s일 때의 두 알고리즘 간 평균 홉 수를 나타낸 그래프이다. 앞에서 예측했듯 단순히 랜덤한 시간 간격만을 INQUIRY나 INQUIRY SCAN에 소비하며 형성된 스캐터넷보다는 적극적으로 트리의 형태까지 고려한 스캐터넷의 장치 간 평균 거리가 더 짧다. 특히 노드 수가 증가할수록 그 양상이 뚜렷하다. 이는 노드 수가 증가하면 증가 할 수 록 형성되는 스캐터넷이 분지 수가 큰 (그림 2)의 형태를 띠게 되기 때문이다.



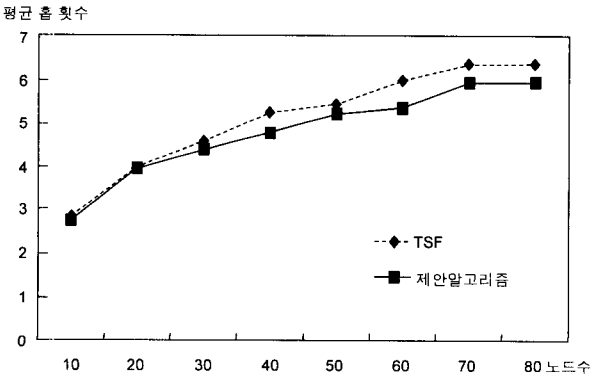
(그림 7) Scan window가 11.25ms인 경우의 지연시간



(그림 8) Scan window가 1.8s인 경우의 지연시간



(그림 9) Scan window가 11.25ms인 경우의 노드 간 평균 홉 수



(그림 10) Scan window가 1.8s인 경우의 노드 간 평균 홉 수

따라서 본 논문이 제시하는 알고리즘은 노드 수가 많은, 즉 다수의 블루투스 장치가 존재하는 곳에서의 스캐터넷 형성 시에 보다 효과적임을 알 수 있다.

5. 결 론

본 논문은 블루투스 명세서상에 기술 되어있는 방법만을 이용하여 추가적인 지연시간을 초래하지 않으면서도 기존의 방법들 보다 장치 간 평균 홉 수가 적은 트리 스캐터넷 형성 알고리즘을 제시하였다. 제시한 알고리즘의 기본 아이디어는 트리 형태의 스캐터넷을 형성해 갈 때 장치 간 거리를 최소로 할 수 있는 트리의 형태를 적극적으로 고려한다는 것이다. 기존에 제시된 알고리즘들은 트리 형태의 스캐터넷을 보장할 뿐 형성 과정에서 트리와 형태 자체가 가지고 있는 이러한 특성을 반영하지 못하였다. 제시한 알고리즘을 검증하기위하여 장치 수와 scan window를 달리해 가며 TSF에 의해 형성된 스캐터넷과 제시한 알고리즘에 의해 형성된 스캐터넷을 전체 네트워크가 형성되기까지의 지연시간과 장치 간 평균 홉 수에 대해 각각 시뮬레이션을 해보았다. 지연시간에서는 차이가 거의 없었고 장치 간 평균 홉 수는 제시한 알고리즘에 의해 형성된 스캐터넷이 더 적었다. 특히 80개 노드 시 scan window가 11.25ms인 경우엔 약 10%, 그리고 scan window가 1.8s인 경우엔 약 6.5%의 평균 홉 수의 개선이 있었다.

앞으로 분지 수가 큰 (그림 2)형태의 트리에 보다 더 가깝도록 동적으로 형성해 가는 방법을 개발해내고 TSF이외에 제시되고 있는 알고리즘들과의 성능 비교 연구가 필요하다. 또한 그래프 이론이 제시하는 여러 위상의 장단점을 검토하여 스캐터넷 형성에의 활용성 여부를 검토하는 것도 중요한 연구 주제이다.

참 고 문 헌

[1] "Bluetooth Extension for ns," http://oss.software.ibm.com/developerworks/open-source/bluehoc/, 2001.

[2] Bluetooth SIG, "Specification of the Bluetooth System ver 1.1," http://www.bluetooth.com/, February, 2001.

[3] J. Bray and C. F. Sturman, 'BLUETOOTH 1.1 Connect Without Cables,' 2nd Ed, Prentice Hall, 2001.

[4] M. Fischermann, A. Hoffmann, D. Rautenbach, L. Szekely and L. Volkmann, "Wiener Index Versus Maximum Degree in Trees," Discrete Applied Mathematics 122, pp. 127-137, 2002.

[5] J. C. Haartsen, "The Bluetooth Radio System," IEEE Personal Communications Magazine, pp.28-36, February, 2000.

[6] E. Horowitz, S. Sahni, D. Mehta, 'FUNDAMENTALS OF

DATA STRUCTURES IN C++', COMPUTER SCIENCE PRESS, 1994.

[7] Ching Law, Amar K. Mehta and Kai-Yeung Siu, "Performance of a New Bluetooth Scatternet Formation Protocol," ACM Symposium on Mobile Ad Hoc Networking and Computing, October, 2001.

[8] "ns-2 Network Simulator," http://www.isi.edu/nsnam/vint/, 2000.

[9] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," Proc. IEEE INFOCOM, April, 2001.

[10] M. Sun, CK. Chang and TH. Lai, "A Self-Routing Topology for Bluetooth Scatternet," Proc. I-SPAN 2002, Manila, Philippines, May, 2002.

[11] G. Tan, A. Miu, J. Guttag and H. Balakrishnan, "Forming Scatternets from Bluetooth Personal Area Networks," MIT-LCS-TR-826, October, 2001.

[12] G. Tan, "Self-organizing Bluetooth Scatternets," MS Thesis, MIT, January, 2002.

[13] G. Tan, "Interconnecting Bluetooth-like Personal Area Networks," 1st Annual Oxygen Workshop, Gloucester, MA, 2001.

[14] C.-K. Toh, 'Ad Hoc Mobile Wireless Networks : Protocols and Systems,' Prentice Hall PTR, 2001.

[15] Zhifang Wang, Robert J. Thomas and Zygmunt Haas, "Bluenet - a New Scatternet Formation Scheme," Proceedings of the 35th Annual Hawaii International Conference on System Sciences, January, 2002.

[16] G. V. Zaruba, S. Basagni and I. Chlamtac, "Bluetrees-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks," ICC 2001, Vol.1, pp.273-277, June, 2001.

강 승 호

e-mail : kinston@caesar.chonnam.ac.kr

1994년 전남대학교 전산학과(이학사)

2003년 전남대학교 대학원 전산학과(이학석사)

2003년~현재 전남대학교 대학원 전산학과 박사과정

관심분야 : 알고리즘, 병렬 및 분산처리, Ad Hoc 네트워크, 바이오인포마틱스

강 대 욱

e-mail : dwkang@chonnam.ac.kr

1982년 전남대학교 계산통계학과(이학사)

1985년 전남대학교 계산통계학과(이학석사)

2000년 영국 Newcastle University 박사과정 수료

1984년~현재 전남대학교 컴퓨터정보학부 교수

관심분야 : Mobile Computing, distributed Computing, Network, Mobile Agent

임 형 석

e-mail : hslim@chonnam.ac.kr

1983년 서울대학교 컴퓨터공학과(공학사)

1985년 한국과학기술원 전산학과(이학석사)

1993년 한국과학기술원 전산학과(공학박사)

1996년~1997년 미국 Purdue 대학 방문교수

1987년~현재 전남대학교 전자컴퓨터정보통신공학부 교수

관심분야 : 알고리즘, 병렬 및 분산처리