하드웨어 구현을 기반으로 한 블루투스 스캐터넷 형성 알고리즘

(A Bluetooth Scatternet Formation Algorithm based on Hardware Implementation)

이 한 욱 † 고 상 근 ††
(Han-Wook Lee) (S. Ken Kauh)

요 약 블루투스는 휴대폰을 비롯한 디지털 디바이스간의 탄력적이고 확장성 있는 무선 ad-hoc 네트워크를 제공할 수 있는 기술로 가능성을 인정받아왔다. 이러한 네트워크가 지원되기 위해서는 블루투스 스캐터넷(Scatternet)은 필수적인 요소이다. 그러므로 블루투스 스캐터넷과 관련되어서 현재까지 다양한 방법론이 제시되고 있다. 하지만 기존의 연구들은 시뮬레이션을 통한 방법론을 제시하는데 그치는 경우가 대부분이며, 스캐터넷의 탄력성과 확장성을 확보하지 못하고 그 복잡성으로 인해 하드웨어 구현에 제약이 있는 경우들이 많다. 본 논문에서는 실제 하드웨어 구현이 용이하고, 스캐터넷의 탄력성과 확장성을 확보할 수 있는 노드 링 스캐터넷(Node Ring Scatternet:NRS) 알고리즘을 제안하였다. 이 알고리즘은 초기 형성과 재형성 부분으로 구성이 되어 있다. 초기 형성과 관련되어 제한적 SEEK/SCAN 알고리즘을 제안하였고, 재형성과 관련되어서는 DIAC 알고리즘과 예약 복구 노드(Reserved Recovery Node) 알고리즘을 제안하였다. 또 실제 시스템 상에서 스캐터넷 알고리즘을 운용하기 위한 SFMP(Scatternet Formation & Management Protocol)을 설계하고, 상용 블루투스 하드웨어로 구현을 하여 20개의 디바이스까지의 스캐터넷 실험을 수행하였다. 실험 결과 기존의 유사 알고리즘에 비해 스캐터넷 형성 시간과 그 확률이 높은 결과를 얻었다.

키워드 : 블루투스, 스캐터넷, 노드 링 스캐터넷 알고리즘, 제한적 SEEK/SCAN 알고리즘, DIAC 알고리즘, 예약 복구 노드 알고리즘

Abstract Bluetooth has been reputed as a wireless ad-hoc networking technology supplying scalable and extensible networks between digital devices. For that kind of networks, scatternet is a most essential part in bluetooth. But past researches on bluetooth scatternet has proposed only possibilities of scatternet algorithm based on simulation results. And many of the researches failed in guaranteeing extensibility and flexibility and had many difficulties in real hardware implementation. In this paper, we proposed node ring scatternet(NRS) algorithm guaranteeing extensible and flexible networks. NRS algorithm is designed for hardware implementation using real commercial bluetooth module. That algorithm is divided into initial formation and reformation. For initial formation, we proposed limited SEEK/SCAN algorithm. For reformation, we proposed DIAC algorithm and Reserved Recovery Node algorithm. And we proposed SFMP(Scatternet Formation & Management Protocol) in protocol stack for real implementation. NRS algorithm is operated in SFMP. Finally, we performed real hardware experiments and evaluated the proposed algorithm. In that experiments, we succeeded in forming scatternet up to 20 nodes. In comparison with other similar algorithm, proposed algorithm have the improvement in scatternet formation delay and success rate.

Key words : Bluetooth, Scatternet, Node Ring Scatternet Algorithm, Limited SEEK/SCAN Algorithm, DIAC Algorithm, Reserved Recovery Node Algorithm

† 학생회원 : 서울대학교 기계항공공학부
equinox2@snu.ac.kr
†† 정 회 원 : 서울대학교 기계항공공학부 교수
kauh@snu.ac.kr
논문접수 : 2004년 1월 7일
심사완료 : 2004년 3월 10일

1. 서 론

블루투스(Bluetooth)는 휴대폰을 비롯한 디지털 디바이스 간의 탄력적이고 확장성 있는 무선 인터페이스를 제공하는 기술로 가능성을 인정받아 왔다. 특히 블루투

스는 피코넷(Piconet)과 스캐터넷(Scatternet)이라는 두 가지 형태의 네트워크를 지원하여, 이를 이용한 ad-hoc 네트워크 형성이 가능하다. 이중 피코넷은 1개의 마스터(Master) 디바이스에 7개 까지의 슬레이브(Slave) 디바이스를 연결할 수 있는 소규모 네트워크이고, 현재의 블루투스 규격에 부합되는 상용 하드웨어에서 안정적으로 지원이 된다.

스캐터넷은 다수의 피코넷을 연결하여 규모가 크고 확장성 있는 네트워크를 구성하는 것이다. 이 스캐터넷은 블루투스가 디지털 디바이스 사이에서 확장성 및 탄력성을 지닌 ad-hoc 네트워크를 지원하기 위해서 필수적인 요소이다. 하지만 그동안 1.1 버전의 규격에서는 스캐터넷과 관련하여 간단한 방법론 정도만 제시한 것에 그쳤으며[1], 최근에 발표된 1.2 버전에서도 스니프(Sniff) 상태를 이용한 피코넷 간의 연결 스케줄에 대한 간단한 언급만 있는 정도이다[2]. 그러므로 현재까지 스캐터넷과 관련되어서는 명확한 방법론이 확정되지 못한 채 다양한 형태의 연구가 진행되어왔다.

블루투스는 특정한 베이스밴드 혹은 MAC의 특성 상 기존에 연구된 일반적인 ad-hoc 네트워킹 알고리즘을 적용하기에는 어려움이 많다[3]. 또 기존의 스캐터넷 관련 연구들은 알고리즘을 제시하고 시뮬레이션을 통한 가능성을 검증하는 정도에 그치는 것이 대부분이다. 이중 그 알고리즘이 복잡하거나 베이스밴드(Baseband) 혹은 MAC의 일부를 변경해야 하는 경우도 있어 하드웨어 구현에 제약이 있는 경우들도 있다. 또 대부분의 기존 연구들은 하드웨어 구현을 위한 구체적인 사항들을 명확하게 제시하지 못하고 있다.

본 논문에서는 실제 하드웨어 구현을 고려하고, 리소스가 제한된 시스템 환경에서도 동작이 가능하도록 그림 1과 같은 형태의 노드 링 스캐터넷(Node Ring Scatternet:NRS) 알고리즘을 제시하였다. 또 NRS는 블루투스의 본질적인 특징인 네트워크의 확장성과 탄력성을 확보하는데 중점을 맞추어 설계하였다. 본 논문의 NRS 알고리즘에서는 초기 스캐터넷 형성과 재형성 관련 알고리즘을 제안하였다. 또 이를 운용하기 위한 프로토콜인 SFMP(Scatternet Formation & Management Protocol)를 설계하여 상위 프로토콜 스택 상에 위치하게 하였다. 이를 기반으로 NRS 알고리즘을 상용 칩셋 하드웨어에 구현하였고, 실험 결과를 도출하였다. 본 하드웨어 구현을 통하여 실제 20개의 블루투스 디바이스의 연결이 안정적으로 동작함을 확인하였다.

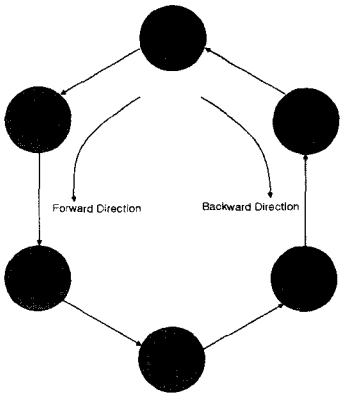

그림 1 링형 스캐터넷의 형태

본 논문의 구성은 다음과 같다. 먼저 2장에서 관련 연구를 설명하고, 3장에서 NRS 알고리즘에 대해 설명한다. 4장에서는 3장에서 제시한 알고리즘을 기반으로 실제 하드웨어 구현을 위한 내용을 다루고, 5장에서는 성능 평가를 위한 실험 결과를 제시한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

스캐터넷에 관련된 연구는 크게 세 가지로 분류할 수 있다. 스캐터넷의 형태에 따른 형성 알고리즘에 관련된 연구[4-12], 라우팅 관련 연구[3], 브릿지 노드 등에서 스케줄링 관련 연구[13] 등이 그것이다.

이중 스캐터넷 형성 알고리즘에 관련되어서는 스캐터넷의 형태에 따라 트리형[4-7],스타형[8], 링형[9-11]으로 나눌 수 있다. 트리형의 대표적인 연구로는 MIT-TSF[4], MIT-BSFA[5,6], Bluetrees[7]가 있다. 트리형은 형태 자체가 확장성이 높으며, 하나의 노드에 연결되는 피코넷 연결을 2개까지로 제한할 수 있다. 또 스캐터넷 내부에서 노드간 형성되는 링크 수를 최소화 할 수 있다는 장점을 지니고 있다. 하지만 트리형은 노드의 개수가 많아질 경우 트리의 루트(Root) 부분에 병목현상이 일어나게 된다. 또 하나 이상의 노드에서 에러가 발생하면 복구 및 재형성 과정이 복잡한 단점을 지니고 있다. 이외에도 노드 사이의 경로가 하나만 존재하므로 결함 극복(Fault Tolerance)면에서 약한 단점을 지니고 있다[9].

BlueStar[8]로 대표되는 스타형은 브릿지 노드를 통해 다수의 피코넷을 연결하는 형태이다. 이 경우 피코넷의 개수가 많아지고, 피코넷 내부에 소속된 노드의 개수가 일정하지 않을 경우 전체 스캐터넷의 형태는 매우 복잡해진다. 결과적으로 노드 사이의 경로가 효율적이지 못하며, 라우팅이 복잡해진다는 단점이 있다.

링형 스캐터넷에 대한 연구는 크게 두 가지로 분류할 수 있다. 피코넷을 기본 단위로 한 피코넷 링의 형태[9,10]와 노드를 기본 단위로 노드 링의 형태[11]이다. 피코넷 링의 형태는 피코넷 사이에 슬레이브-슬레이브

브릿지(Bridge)를 통해 다수의 피코넷을 링형으로 연결한 것이다.

그런데 위와 같은 형태들은 스캐터넷의 기본 단위를 피코넷으로 하고 있다. 피코넷을 기본으로 형성된 스캐터넷의 경우 효율성은 높을 수 있지만, 피코넷 사이의 분리 및 통합, 역할 교환(Role Switching) 등 초기 형성 과정 혹은 재형성 과정에서 그 알고리즘이 복잡한 경우들이 많다. 특히 마스터 노드의 스캐터넷 이탈 시 그 재형성 과정이 매우 복잡한 단점이 있다.

반면 [11]에서 제시한 링형 알고리즘은 그림 1과 같이 노드를 기본 단위로 하고 있다. 이 알고리즘에서는 모든 노드가 마스터/슬레이브 릴레이(Relay)로 동작한다. 즉 스캐터넷 내부의 모든 피코넷은 마스터-슬레이브의 1 대 1 연결로 구성된다. 그러므로 스캐터넷 내부에 마스터, 슬레이브, 브릿지 혹은 리더(Leader) 등과 같은 역할 구분이 없으며, 모든 노드의 역할이 동등한 진정한 의미의 분산 네트워크이다. 특히 노드가 추가 혹은 제거되는 동적 상황에서도 피코넷 재형성 과정이 없으므로 시간 지연이 짧고 그 알고리즘이 간단하다.

그런데 이상에서 언급한 기존 연구들은 알고리즘을 제시하고 시뮬레이션으로 이를 검증한 차원에서 그치는 경우가 대부분이다. 또 실제 하드웨어 구현에 제약이 많거나, 이를 위한 구체적인사항이 명확하게 제시되지 못하고 있다.

3. 연구 내용

본 논문에서는 그림 1과 같이 노드를 기반으로 하는 링형 스캐터넷을 구성하였다. 2장에서 언급했듯이 이러한 형태는 알고리즘의 간략성으로 인해 하드웨어 구현상 제약이 없고, 리소스가 제한된 환경에서도 동작이 가능하다. 또 노드 추가 혹은 제거 시 피코넷 재형성 과정이 없으므로, 그 알고리즘이 간단하고 안정적이다. 그러므로 간단한 알고리즘으로 네트워크의 확장이 용이하고, 동적인 상황에서도 탄력성을 확보할 수 있다.

이러한 형태의 스캐터넷 관련 연구는 [11]에서 선행되었다. 하지만 [11]에서 제시된 알고리즘은 초기 스캐터넷 형성 시 시간 지연이 크고, 재형성에 관련된 언급이 없다. 또 알고리즘 자체의 간단한 방법론만을 제시할 뿐 실제 구현에 적용될 정도로 명확하지 못하다.

본 논문에서는 [11]에서 제안된 초기 형성 알고리즘을 수정하고, 재형성 알고리즘을 추가하여 노드 링 스캐터넷(Node Ring Scatternet:NRS)라고 명명하였다. 특히 기존 연구와 달리 이 알고리즘을 하드웨어 구현이 가능하도록 구체화하여 프로토콜 스택 상의 하나의 프로토콜로 구현하였다.

본 논문에서 제안한 NRS 알고리즘은 초기 형성 알고리즘과 재형성 알고리즘으로 나누어진다. 초기 형성 알고리즘은 기존의 SEEK/SCAN 알고리즘을 NRS에 맞게 제약조건을 둔 제한적 SEEK/SCAN 알고리즘을 제안하였다. 또 재형성 알고리즘과 관련하여 노드 제거 시 복구 알고리즘으로 DIAC 알고리즘과 예약 복구 노드 알고리즘(Reserved Recovery Node Algorithm)을 제안하였다. 위 알고리즘들은 모두 시간적 지연을 줄이고, 동작의 안정성을 높이기 위한 방법으로 제안되었다.

3.1 초기 형성 알고리즘

3.1.1 초기 형성 알고리즘 개요

일반적으로 스캐터넷의 초기 형성 시 노드 사이의 연결은 SEEK/SCAN 알고리즘을 따르는 경우들이 많다 [5,6,8,10,11]. 그런데 SEEK/SCAN 알고리즘은 노드 사이의 연결이 분산적이고 자율적으로 이루어지므로, 그림 1과 같은 NRS 형태에 적용하였을 경우 연결 과정중 소규모 링이 여러 개 형성되어 하나의 링으로 통합되는데 그 과정이 복잡해지고, 긴 시간 지연을 유발할 수 있다. [11]에서는 이러한 문제점을 해결하기 위해 NODE_ID 알고리즘과 HEAD_SEEKSCAN 알고리즘을 제시하였다. 두 알고리즘은 모두 링형 네트워크를 구성하기 위한 전단계로 선형 구조를 형성하는 것인데, 이중 HEAD_SEEKSCAN 알고리즘의 경우 다른 선형 구조에 소속된 노드 개수 정보를 알아야하므로, 링크 형성을 통한 ACL 데이타 과정이 필요하다. 또 재형성을 위해 HEAD 노드가 SEEK/SCAN 동작을 모두 수행해야 하므로 HEAD 노드에서 오버 헤드가 걸리고, 라우팅 시 병목현상이 발생할 수 있다. 반면 NODE_ID 알고리즘은 Inquiry 과정을 통해 얻어지는 타 노드의 BD_ADDR 정보를 기반으로 선형구조를 형성하므로, 링크 연결을 통한 사전 정보 교환 과정이 없어 시간적 지연이 단축된다. 또 재형성을 위한 SEEK/SCAN 동작이 HEAD와 TAIL에서 분산되어 이루어진다. 그러므로 본 연구에서는 NODE_ID 알고리즘을 기반으로 하였다.

또 본 알고리즘에서는 스캐터넷 형성 과정에 필요한 6가지 상태를 설정하였고, 각 노드들은 그림 2와 같은 상태 천이를 거쳐 스캐터넷을 형성하게 된다. 각 상태들이 천이되는 조건은 아래와 같다.

1. 모든 노드는 초기에 INIT 상태로 있고, 타 노드에 대한 사전 정보가 없다. INIT 상태에서는 SEEK/SCAN 알고리즘을 반복한다.
2. INIT 상태에서 SEEK 동작 수행 중에 다른 노드를 발견하고 연결이 이루어지면 HEAD 상태로 천이되며, SCAN 동작만 주기적으로 수행한다.
3. INIT 상태에서 SCAN 동작 수행 중에 다른 노드로부터 연결이 요청된 후 연결이 이루어지면 TAIL 상태로 천이되며, SEEK 동작만 주기적으로 수행한다.

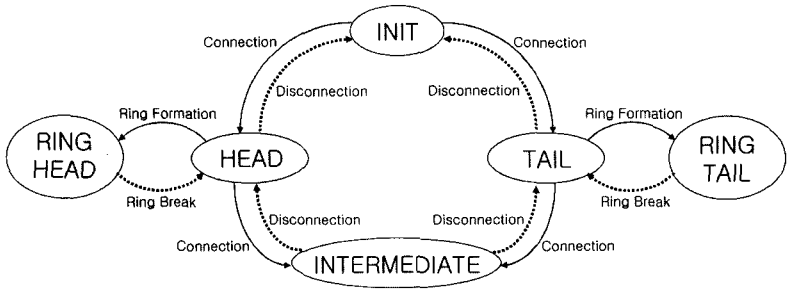

그림 2 NRS 알고리즘의 노드 상태 천이도

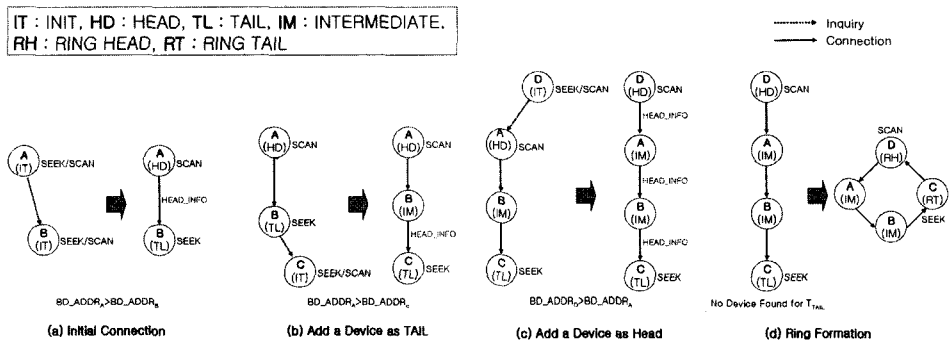

그림 3 스캐터넷 형성 과정의 예

이때 TAIL 노드는 반드시 HEAD의 정보를 넘겨받아야 한다.

4. HEAD 혹은 TAIL 상태에서 추가적인 연결이 이루어지면 모두 INTERMEDIATE 상태로 천이되고, 아무 동작을 수행하지 않는다.
5. 선형 구조에서 링이 형성되면 HEAD 상태 노드는 RING HEAD 상태로, TAIL 상태 노드는 RING TAIL 상태로 천이된다.

그림 3에는 스캐터넷 형성 예를 나타내었다. 모든 노드들은 NODE_ID 알고리즘을 기반으로 BD_ADDR이 자신의 값보다 작은 노드에게만 연결을 할 수 있다. 그림 3의 (a)는 INIT 상태의 두 개 노드가 있는 경우이다. 이때 A의 BD_ADDR이 B의 값보다 크다. 그러므로 SEEK/SCAN 알고리즘 실행 중 B가 A를 발견하더라도 NODE_ID 알고리즘에 의해 연결 요청을 하지 못한다. 결국 A가 B를 발견하고 연결이 되면 A는 HEAD 상태, B는 TAIL 상태로 천이된다. 이때 A는 B에게 BD_ADDR과 클럭 오프셋(Clcok Offset)[1]으로 구성된 자신의 정보를 전달한다. 이후 A는 SCAN 동작만을, B는 SEEK 동작만을 주기적으로 반복한다.

(b),(c)는 새로운 노드가 추가되는 경우인데, (b)는 HEAD인 A의 BD_ADDR보다 작은 값을 지닌 노드 C가 추가되는 경우이다. 이때 B는 주기적인 SEEK 동작을 통해 C를 발견하고, 저장된 HEAD A의 BD_ADDR과 비교하여 그 값이 작으므로 연결을 한다. (c)의 경우는 HEAD인 A의 BD_ADDR보다 큰 값을 지닌 노드 D가 추가되는 경우로, D는 자신의 BD_ADDR보다 작은 HEAD A에 연결 요청을 한 후 새로운 HEAD가 된다. 이때 새로운 HEAD D의 정보는 A, B를 거쳐 TAIL인 C에게 전달된다.

이러한 선형 구조는 TAIL인 C에서 T_{TAIL}동안 HEAD D외의 새로운 디바이스를 발견하지 못하면 (d)와 같은 링형이 형성된다. 이후 C는 RING TAIL로 천이되어 SEEK 동작을, D는 RING HEADER로 천이되어 SCAN 동작을 주기적으로 반복하여 새로운 노드의 추가 연결을 준비한다.

3.1.2 노드 연결 알고리즘

3.1.2.1 기존의 SEEK/SCAN 알고리즘 고찰

그림 3과 같은 과정으로 스캐터넷이 형성되기 위해서는 각 노드 사이의 분산적이고 자율적인 노드 연결 알고리즘이 필요하다. 기존의 다수의 연구들에서는 SEEK/

1) BD_ADDR과 클럭 오프셋(Clock Offset)은 Page를 수행하기 위한 HCI 명령인 Create_Connection의 입력 인자가 된다. 즉 위 두 정보를 가지고 있으면 Inquiry 과정을 거치지 않고 Page를 수행할 수 있다.

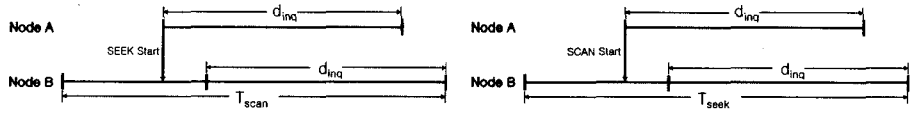

그림 4 SEEK/SCAN 알고리즘을 통한 노드 간의 연결 형태

SCAN 알고리즘을 기반으로 하여 노드 간의 연결에 적용하였다[5,6,8,10,11]. SEEK/SCAN 알고리즘이란 스캐터넷 형성 초기의 각 노드들이 난수 확률에 근거하여 SEEK 혹은 SCAN 동작을 결정하여 타 노드와의 연결이 이루어질 때까지 반복하는 것이다.

그런데 이러한 SEEK/SCAN 알고리즘의 경우 동작이 반복되기 위해서는 각 동작에 대한 시간 제한이 필요하다. 본 논문에서는 이 시간 제한을 각각 T_{SEEK}, T_{SCAN}으로 정하였다. 또 각 동작이 1회 이루어지는 시간 단위를 라운드(Round)라고 하였다. 그런데 여기서 T_{SEEK}, T_{SCAN}은 노드간의 연결 확률 및 시간 지연을 결정짓는 매우 중요한 인자로 예상되지만, 기존 대부분의 연구들은 이 부분에 대한 언급이 없는 상태이다.

본 논문에서는 T_{SEEK}, T_{SCAN}의 영향을 분석하기 그림 4와 같이 SEEK/SCAN 알고리즘으로 노드가 연결되는 두 가지 경우를 고려하였다. 블루투스는 초기에 노드의 연결을 위해서는 Inquiry 과정을 통해 타 디바이스에 대한 정보를 얻어야만 한다. 본 논문에서는 이 Inquiry 동작을 통해 타 노드의 정보를 얻는데 필요한 시간을 d_{inq}라고 한다.

(a)의 경우는 노드 B가 현재의 라운드에서 SCAN 동작을 수행하고 있고, 노드 A는 SEEK 동작을 시작하는 경우이다. 이때 A가 B의 T_{SCAN}이 d_{inq}이상 남은 상태에서 SEEK 동작을 수행하면 연결이 이루어지게 된다. 반면 (b)의 경우는 B가 현재 라운드에서 SEEK 동작을 수행 중에 있고, A가 SCAN 동작을 시작하는 경우이다. 이 경우에도 A는 B의 T_{SEEK} 지속 시간이 d_{inq}이상 남은 상태에서 SCAN 동작을 수행하면 연결이 이루어지게 된다. 여기서 노드가 새로운 라운드가 시작될 때 SEEK 혹은 SCAN 동작을 하게 될 확률을 각각 P_{SEEK}, P_{SCAN}이라 하면 아래와 같은 관계가 성립한다.

$$P_{SEEK}+P_{SCAN}=1$$

또 그림 4의 두 경우에 대해 연결이 이루어질 확률 P_{CON}을 표현하면 다음과 같다.

$$P_{CON}=P_{SEEK}\frac{T_{SCAN}-d_{inq}}{T_{SCAN}}P_{SCAN}+P_{SCAN}\frac{T_{SEEK}-d_{inq}}{T_{SEEK}}P_{SEEK}$$

위 두 개의 식을 정리하면 아래와 같은 결과가 얻어진다.

$$P_{CON}=P_{SCAN}(1-P_{SCAN})(2-d_{inq}(\frac{1}{T_{SCAN}}+\frac{1}{T_{SEEK}}))$$

또 T_{SEEK}과 T_{SCAN}은 모두 d_{inq}보다 커야하고, 위 식에서 P_{CON}을 최대화하기 위해서는 아래와 같이 조건을 정리할 수 있다.

$$P_{SCAN}=P_{SEEK}=0.5$$

$$d_{inq}(\frac{1}{T_{SCAN}}+\frac{1}{T_{SEEK}})<1$$

$$T_{SEEK}, T_{SCAN} > d_{inq}$$

위 식에 따르면 d_{inq}는 고정된 상수로 놓을 수 있으므로 T_{SEEK}과 T_{SCAN}이 SEEK/SCAN 알고리즘의 연결 확률을 결정짓는 인자임을 알 수 있다. 즉 T_{SEEK}과 T_{SCAN}을 작게 설정하면 P_{CON}값이 작아지므로 연결이 되기까지 여러 번의 라운드를 반복해야할 가능성이 높아져 시간 지연이 길어지게 된다. 반면 T_{SEEK}과 T_{SCAN}을 크게 설정하면 P_{CON}값이 커지므로 몇 번의 라운드 안에 노드 간의 연결이 이루어질 확률이 높다. 다만 T_{SEEK}과 T_{SCAN}을 크게 설정하면 한 번의 라운드 지속 시간이 길게 되므로, 한번의 라운드에서 연결이 이루어지지 못하면 다음 라운드까지 긴 시간 지연을 초래한다.

본 논문에서는 5.1.1에서 T_{SEEK}과 T_{SCAN}에 따른 스캐터넷 형성 시간을 실험적으로 측정하였고, 최적화된 T_{SEEK}과 T_{SCAN} 값을 제시하였다.

3.1.2.2 NRS를 위한 제한적 SEEK/SCAN 알고리즘의 제안

[11]에서는 NODE_ID 알고리즘의 노드 간 연결을 위해 일반적인 SEEK/SCAN 알고리즘을 사용하고 있다. 하지만 NODE_ID 알고리즘은 BD_ADDR의 크기에 의해 연결이 결정되므로, SEEK/SCAN 알고리즘을 그대로 적용하면 아래의 세 가지 형태의 시간 지연이 발생할 수 있다. 두 개의 노드 A,B가 있고, A의 BD_ADDR이 B보다 큰 경우라고 가정하면

1. 노드 A,B가 동시에 SEEK 동작 혹은 SCAN 동작을 수행하는 경우
2. A가 SCAN 상태에 있는 경우
3. B가 SEEK 상태에 있는 경우

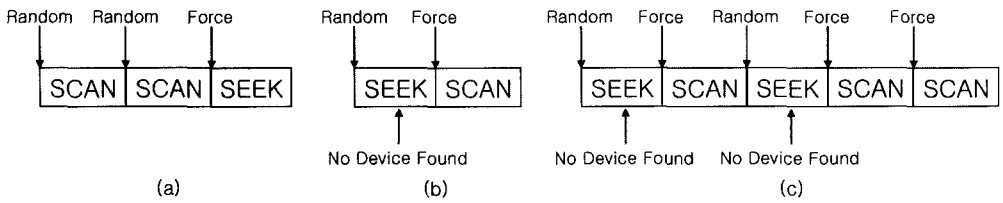

그림 5 제한적 SEEK/SCAN 알고리즘

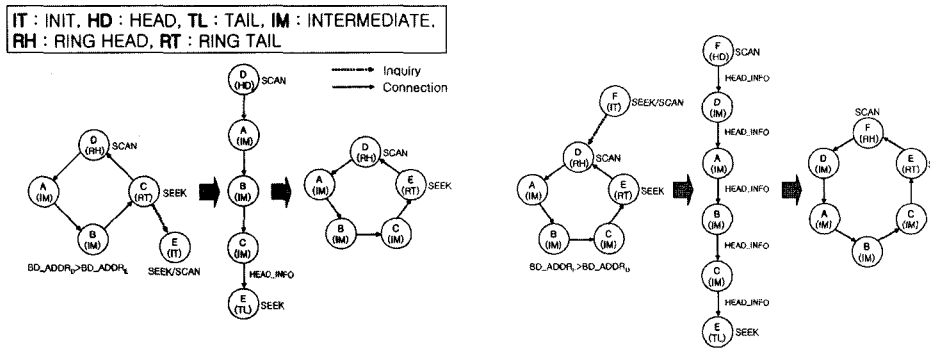

그림 6 노드 추가 알고리즘

1의 경우는 SEEK/SCAN 알고리즘이 난수 확률에 의해 동작을 결정하므로 근본적으로 가지고 있는 문제점이다. 하지만 2,3의 경우는 NODE_ID 알고리즘으로 인해 발생하는 것이다. 즉 NODE_ID 알고리즘에 의해 빠른 시간 내에 연결이 이루어지기 위해서는 A는 SEEK 동작을 B는 SCAN 동작을 수행해야 한다. 그러나 난수 확률에 의해 반대의 동작, 즉 A는 SCAN 동작, B는 SEEK 동작을 수행하게 되면 그 라운드는 시간 지연으로 이어진다. 특히 T_{SEEK}과 T_{SCAN}이 길수록 라운드 지속 시간이 길어져 그만큼 시간 지연이 커지게 된다.

본 논문에서는 위와 같은 문제점을 해결하기 위해 NODE_ID 알고리즘에 맞도록 기존의 SEEK/SCAN 알고리즘에 아래와 같은 제약 조건을 추가한 제한적 SEEK/ SCAN 알고리즘(Limited SEEK/SCAN Algorithm)을 제시하였다. 제한적 SEEK/SCAN 알고리즘은 그림 5에 나타내었고, 각 그림에 따른 제약 조건은 다음과 같다.

(a) SCAN 동작은 연속적으로 3번의 라운드 이상 반복하지 않는다. 즉 난수 확률에 근거하여 2번의 라운드에서 연속적으로 SCAN 동작을 하였다면, 다음 라운드에서는 무조건 SEEK 동작을 수행한다.

(b) SEEK 동작 수행 중에 노드를 발견하지 못하거나, 혹은 발견된 노드 중에 연결할 수 있는 노드가 없다면[2], 다음 라운드에서는 무조건 SCAN 동작을 수행한다.

(c) (b)의 과정에서 SCAN 동작 이후, 다음 라운드에서 SEEK 동작을 수행하고, 또 노드를 발견하지 못하거나 발견된 노드 중 연결할 수 있는 것이 없다면, 다음 두 번의 라운드를 연속적으로 SCAN 동작을 수행한다.

이상과 같은 제한적 SEEK/SCAN 알고리즘을 NRS에 적용하여 기존의 SEEK/SCAN 알고리즘과의 스캐터넷 형성 시간 및 형성 확률의 차이를 5.1.2에서 실험적으로 측정하였다.

3.2 재형성 알고리즘

스캐터넷 재형성이란 스캐터넷에 새로운 노드가 추가되거나, 스캐터넷 내부의 노드 중 하나 이상에서 에러가 발생하거나 제거되었을 때 전체 스캐터넷을 재구성하는 것이다. 즉 재형성 알고리즘은 노드 추가 알고리즘과 복구 알고리즘으로 나누어 진다.

3.2.1 노드 추가 알고리즘

그림 6에는 노드 추가 알고리즘을 나타내었다. 추가 알고리즘의 기본 구조는 새로운 노드가 발견되면 링을

2) Inquiry 과정에서 발견된 노드를 연결하지 못하는 경우는 두 가지이다. 첫째는 BD_ADDR이 자신의 값보다 큰 경우이고, 둘째는 발견된 노드가 디바이스 필터에 등록된 경우이다. 디바이스 필터에 대한 내용은 4.1에서 다루기로 한다.

분리하여 선형 구조를 만들고, 노드를 추가한 후에 다시 링을 형성하는 형태이다.

그림 6의 (a)의 경우는 RING HEAD인 노드 D보다 BD_ADDR이 작은 노드 E가 추가되는 경우이다. RING TAIL인 C는 주기적인 SEEK 동작 중 E를 발견하고 NRS 지원 여부를 확인한 후, D와 연결을 종료하고 E와 연결한 선형 구조를 만든다. 이 상태에서 E는 새로운 TAIL이 되고, C는 INTERMEDIATE 상태로 천이되어 E에게 HEAD의 정보를 넘겨준다. 이후 E는 HEAD의 정보를 이용하여 D와 연결 후 링을 재형성한다.

(b)는 RING HEAD인 D보다 큰 BD_ADDR을 지닌 노드 F가 추가되는 경우이다. 이때 F는 SCAN 상태의 RING HEAD D를 발견하고 연결하게 된다. 이때 D는 일시적으로 3개의 피코넷에 연결된 상태가 되고, 이후 E와의 연결을 종료하여 선형 구조를 형성한다. F는 새로운 HEAD가 되었으므로 자신의 정보를 D,A,B,C를 거쳐 TAIL인 E에게 전달한다. E는 이 정보를 이용하여 HEAD F와 연결하여 링을 재형성한다.

이상과 같은 재형성 알고리즘은 5.2에서 하드웨어 실험을 통해 재형성 시간 지연을 측정하였다.

3.2.2 복구 알고리즘

3.2.2.1 복구 알고리즘을 위한 기본 사항

스캐터넷을 구성하는 노드들은 여러 가지 이유로 디바이스간의 연결이 비정상적으로 종료되는 경우가 있다. 본 논문에서는 링크가 종료되는 경우를 그 심각도에 따라 4개의 레벨(Level)로 분류하였다.

레벨 1 : 노드 하나 이상에서 링크 불안정 등으로 인해 연결이 비정상적으로 종료되었지만 재연결이 가능한 경우

레벨 2 : 노드 하나에서 에러가 발생하거나 전원 공급이 중단되어 연결이 비정상적으로 종료되고 재연결이 불가능한 경우

레벨 3 : 연속적이지 않은 2개 이상의 노드에서 에러가 발생하거나 전원 공급이 중단되어 연결이 비정상적으로 종료되고 재연결이 불가능한 경우

레벨 4 : 연속적인 2개 이상의 노드에서 에러가 발생하거나 전원 공급이 중단되어 연결이 비정상적으로 종료되고 재연결이 불가능한 경우

또 본 논문에서는 제안하는 복구 알고리즘은 세 가지를 기본 원칙으로 한다. 첫째 연결의 복구 주체는 마스터이며 복구 방향은 순방향(Forward Direction)[3]으로 일어난다. 즉 NRS에서 노드 사이 하나의 연결은 바로 하나의 피코넷이고, 그 연결이 종료되었을 경우 그 피코넷의 마스터만 복구 알고리즘을 수행할 수 있다. 둘째 하나의 피코넷 내에서 연결이 이루어졌던 슬레이브에 대한 정보는 마스터가 저장하고 있다. 그러므로 노드 간의 연결이 종료되면 마스터는 이 정보를 이용하여 Inquiry 과정 없이 바로 Page 과정으로 연결을 복구할 수 있다. 셋째 본 논문에서 제안하는 복구 알고리즘은 레벨 1~4의 경우에 적용 가능하다.

3.2.2.1 NRS를 위한 복구 알고리즘의 제안

레벨 1의 경우는 연결이 종료되어도 양쪽 노드는 정상인 상태이므로, 마스터가 저장된 정보를 이용하여 페이지 과정의 반복으로 간단하게 연결을 복구할 수 있다. 하지만 레벨 2,3,4의 경우는 하나 이상의 노드에서 문제가 생긴 것이므로, 이웃 노드로 복구해야 한다. 본 논문에서는 노드 이상 레벨에 따라 DIAC 알고리즘과 예약 복구 노드(Reserved Recovery Node) 알고리즘을 선택적으로 적용하였다.

DIAC(Dedicated Inquiry Access Code)를 이용한 방법은 기존의 연구에서 소개된 바가 있다[4,9]. 현재 블루투스 규격에는 63개의 DIAC가 있으며, 이를 이용하면 특정 디바이스만을 선택적으로 검색하는 것이 가능하다[1,2]. 본 논문에서는 이러한 DIAC를 NRS에 적용하여 DIAC 알고리즘이라고 명명하였다.

그림 7의 (a)에서는 노드 A가 스캐터넷을 이탈하게 된다. 이후 D에서는 비정상적인 링크 종료가 통보되고, D는 레벨 1의 상황으로 판단하여 Page 작업을 통한 A로의 재연결을 N_{RETRY} 시도한다. N_{RETRY}동안 링크가 복구되지 못하면 Inquiry를 통해 새롭게 복구할 노드를 찾는다. 이때의 Inquiry는 DIAC를 이용하여 이루어진다. 한편 B는 A와 링크가 비정상적으로 종료되면 같은 DIAC로 Inquiry Scan 상태를 유지한다. 결국 D는 B를 발견하고 연결을 하게 되어 스캐터넷이 복구된다. 만약 이 과정에서 GIAC(Gerneral Inquiry Access Code)를 사용하게 되면 RING TAIL인 E가 B를 발견하고 연결하여 스캐터넷의 형태가 비정상적으로 될 수 있다. 연속된 두 개 이상의 디바이스가 동시에 제거되는 레벨 4의 경우에도 DIAC 알고리즘은 적용될 수 있다.

하지만 DIAC 알고리즘은 레벨 3의 경우에 취약한 약점이 예상된다. 그림 (a)에서 연속하지 않은 노드 A와 C가 동시에 스캐터넷을 이탈한 경우를 생각하자. 이후 B와 E는 DIAC로 Inquiry Scan 상태로 들어가고, 동시에 D와 B는 DIAC로 Inquiry를 수행한다. 이때 D의 경우 B를 발견해야 하지만 E를 발견하고 연결을 할 수도 있다. 이 경우 D,E,F 사이에서만 NRS가 형성되고 B는 고립된 노드가 된다. 더구나 B는 DIAC를 기반으로 Inquiry와 Inquiry Scan을 자체적으로 반복하므로 고립 상태가 지속되는 오류가 발생한다. 이외에도 DIAC 알

3) 그림 1과 같이 순방향(Forward Direction)은 마스터에서 슬레이브 방향이다.

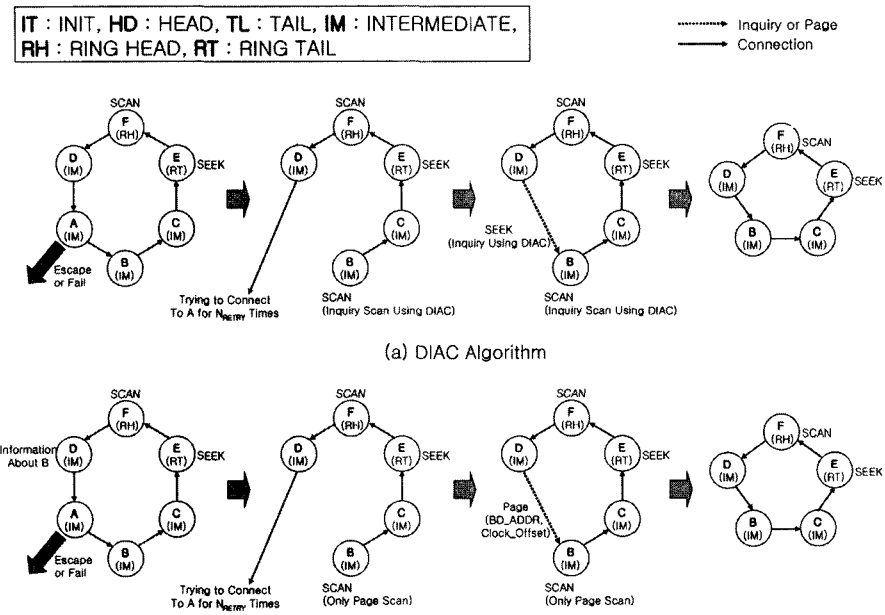

그림 7 복구 알고리즘

고리즘은 Inquiry 과정이 포함되므로 복구 과정의 시간 지연이 크다는 단점이 있다.

본 논문에서 DIAC의 단점을 보완한 예약 복구 노드(Reserved Recovery Node:RRN) 알고리즘을 제안하였다. 이 알고리즘에서 각 노드들은 순방향으로 2홉(Hop) 떨어진 디바이스의 정보를 사전에 저장[4]하고 있다. 즉 각각의 노드들은 이웃 노드의 이상 시 복구하기 위한 노드를 예약하고 있다.

그림 7의 (b)를 보면 D는 예약된 복구 노드인 B의 정보를 사전에 저장하고 있다. 노드 A가 스캐터넷을 이탈한 후 N_{RETRY}동안 A와의 재연결을 시도한 후 복구되지 못하면, 저장된 정보를 바탕으로 노드 B에게 Page를 통해 연결하여 스캐터넷을 복구할 수 있다. RRN 알고리즘은 DIAC 알고리즘에 비해 Inquiry 과정이 없으므로 복구 시간 지연이 짧다는 장점이 있다. 또 레벨 3과 같이 두 개 이상의 노드에 이상이 생긴 경우에도 각 노드들이 복구 노드를 예약하고 있어 안정적으로 스캐터넷이 재형성 된다.

하지만 레벨 4와 같이 연속되는 두 개 이상의 노드가 스캐터넷을 이탈할 경우 복구를 위해서는 순방향으로 3홉 이상의 노드에 대한 정보를 지니고 있어야 한다. 이 경우에는 교환되는 정보량이 많아지고 그 과정도 복잡하게 된다.

본 논문에서는 레벨 2,3의 경우에는 RRN 알고리즘을 적용하고, 레벨 4의 경우에는 DIAC 알고리즘을 적용하기로 하였다. 즉 복구 과정에서 RRN 알고리즘을 먼저 사용하고, 이 방법으로 복구가 되지 못하면 레벨 4의 경우로 판단하여 DIAC 알고리즘을 적용한다. 5.2에는 본 알고리즘을 이용한 복구 실험 결과를 나타내었다.

4. 스캐터넷의 하드웨어 구현

본 논문에서는 스캐터넷을 실제 블루투스 하드웨어로 구현을 하기 위해 새로운 프로토콜을 설계하였으며, PC를 호스트로 하는 시스템을 구성하였다.

4.1 프로토콜의 설계

현재의 블루투스 규격에서는 베이스밴드나 링크 매니져의 하위 계층 프로토콜에서 스캐터넷을 형성을 위해 스니프(Sniff) 모드 등을 이용한 피코넷 간의 스케쥴링을 제공한다[2]. 하지만 그 형성 알고리즘을 현재의 블루투스 규격으로 구현하기 위해서는 별도의 상위 프로토콜의 설계가 필수적이다. 본 논문에서는 그림 8과 같이 스캐터넷 형성과 관련되어 SFMP(Scatternet Formation & Management Protocol)을 설계하여 L2CAP 상위에 위치하게 하였다. SFMP를 통해 본 논문에서 제

4) 순방향으로 2홉 떨어진 정보의 획득을 위해 이웃 노드로 4.1의 SFMP_NeighborInfoRequest PDU를 전달하면, 이웃 노드는 저장된 자신의 이웃 노드 정보를 SFMP_NeighborInfoResponse PDU를 통해 제공한다.

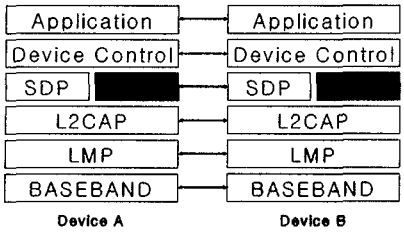

그림 8 하드웨어 구현을 위한 프로토콜의 설계

표 1 SFMP Control PDU의 종류

PDU ID	PDU
0x01	SFMP_ErrorResponse
0x02	SFMP_ConnectionRequest
0x03	SFMP_ConnectionResponse
0x04	SFMP_PutHeadInfo
0x05	SFMP_GetHeadInfo
0x07	SFMP_NeighborInfoRequest
0x08	SFMP_NeighborInfoResponse
0x09	SFMP_DisconnectionRequest
0x0a	SFMP_DisconnectionResponse

안된 NRS 알고리즘을 운용하게 된다.

또 SFMP의 패킷을 그림 9와 같이 설계하였다. SFMP 패킷은 L2CAP의 페이로드로 전송이 되고, 컨트롤 데이타 패킷과 라우팅 데이타 패킷으로 구분이 된다. 컨트롤 데이타 패킷은 SFMP의 운용 상 필요한 컨트롤 PDU를 교환하기 위한 것으로, 인접한 디바이스 사이에서만 교환되는 단일 홉(Hop) 패킷이다. 컨트롤 PDU의 종류는 표 1에 나타내었다. 반면 라우팅 데이타 패킷은 사용자 데이타들이 전송되는 것으로 목적지와 출발지의 BD_ADDR값을 포함하고 있으므로, 이를 이용해 라우팅이 이루어진다.

또 SFMP에는 디바이스 필터를 설계하여 노드 연결 시간을 단축하도록 하였다. 그림 10과 같이 노드의 연결 과정에는 SDP를 통한 서비스 검색 과정을 거치게 된다. 이 과정에서 발견된 노드로부터 SFMP를 기반으로 한 NRS의 지원 여부를 확인하게 된다. 만약 발견된 노드가 NRS에서 지원하지 않게 되면 이 노드의 정보는

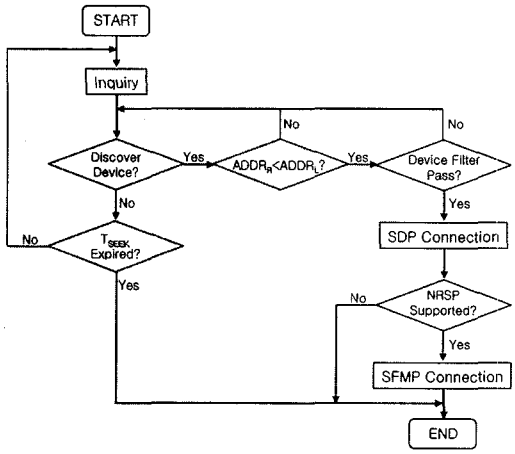

그림 10 SFMP 연결 과정

디바이스 필터 내에 저장이 된다. 이후 SEEK 동작을 반복하는 과정에서 검색된 노드 중 디바이스 필터에 등록된 노드가 있으면 그 노드는 무시된다. 이렇게 하면 NRS가 지원되지 않는 노드에게 서비스 검색 과정을 반복적으로 수행하는 현상을 막을 수 있어 불필요한 시간 지연을 줄일 수 있다.

4.2 하드웨어 및 시스템 설계

본 논문에서는 스캐터넷을 그림 11과 같은 시스템으로 구현하였다. PC를 호스트로 하는 호스트-호스트 컨트롤러 시스템을 구현하였으며, 호스트 컨트롤러는 그림

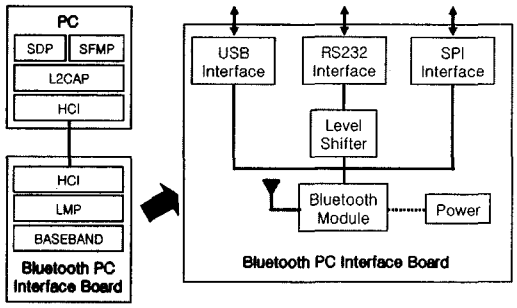

그림 11 하드웨어 시스템의 구조

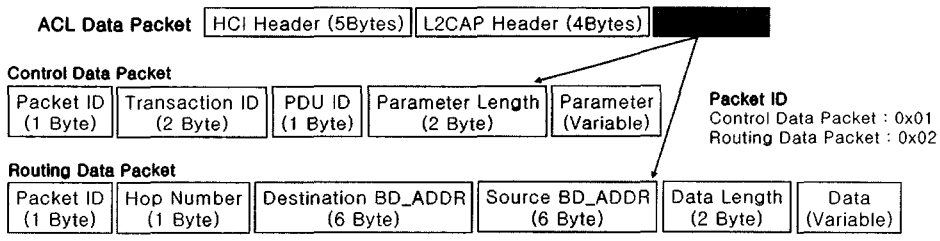

그림 9 SFMP의 패킷 포맷

11과 같이 PC 인터페이스 보드를 제작하였다. 이 보드에는 CSR사의 BlueCore2-External 블루투스 칩셋이 내장된 삼성전기의 BlueSEM-CII(Class2) 모듈이 사용되었다. 현재 다수의 상용 블루투스 칩셋은 스캐터넷 지원에 제약이 많은 경우가 대부분이다. 이 칩셋의 경우 2개의 피코넷으로 구성된 스캐터넷을 안정적으로 지원한다[14].

5. 스캐터넷 알고리즘 및 실험적 성능 평가

본 논문에서는 3장에서 제시한 NRS 알고리즘을 4장에서 설계한 상위 프로토콜 및 하드웨어 시스템에 구현하여 실제 실험을 수행하였다.

5.1 스캐터넷 초기 형성 실험

5.1.1 T_{SEEK}과 T_{SCAN}에 따른 스캐터넷 형성 시간 측정 실험

3.1.2.1에서 T_{SEEK}과 T_{SCAN}은 초기 스캐터넷 형성 시간에 영향을 끼치는 인자임을 언급했다. 또 노드간의 연결 확률을 높이기 위한 T_{SEEK}과 T_{SCAN}의 조건을 제시하였다. 먼저 T_{SEEK}의 조건을 찾기 위해 Inquiry 실험을 수행하여 그림 12와 같은 결과를 얻었다. 실험은 하나의 Inquiry를 수행하는 노드를 중심으로 1m 거리에 방사형으로 Inquiry Scan 상태의 노드를 배치한 상태에서 이루어졌다. 또 Inquiry 시간 간격[5]을 변경시키면서 노드 1개가 발견될 확률을 측정하였다. 그림 12의 결과를 보면 Inquiry 시간 간격이 2.56초일 경우에 50% 미만의 낮은 발견확률을 보였고, 3.84초 이상이 되어야만 안정적인 동작을 할 수 있다.

이제 T_{SEEK}과 T_{SCAN}에 따른 스캐터넷 형성 시간을 측정해 보았다. 스캐터넷 형성 시간은 5개의 노드들이 모두 INIT 상태에서 시작되어 5개의 노드가 모두 그림

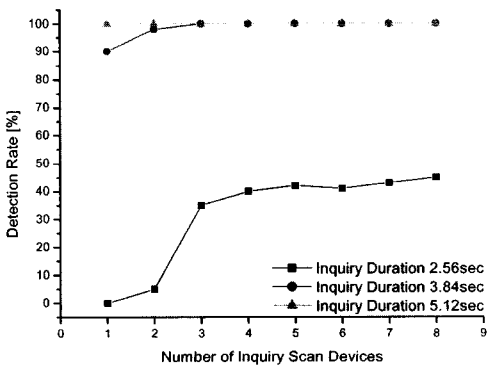

그림 12 Inquiry 실험 결과

5) 블루투스 규격 상 Inquiry 시간 간격은 1.28초의 배수로 설정하게 되어 있다[1,2].

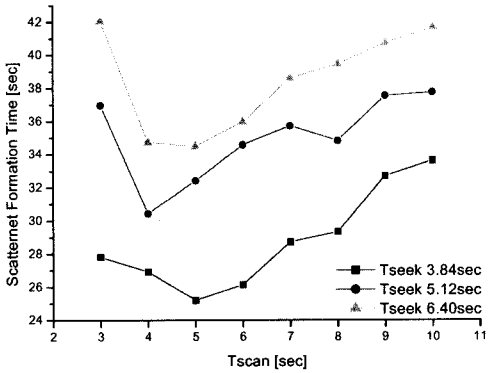

그림 13 T_{SEEK}, T_{SCAN}에 따른 스캐터넷 형성 실험 결과

3의 HEADER-TAIL 선형 관계를 완성할 때까지의 시간을 측정하였다. 그림 13의 결과에 따르면 T_{SEEK}과 T_{SCAN}의 값이 증가할수록 스캐터넷 형성 시간 지연이 커지는 결과를 얻었다. 특히 T_{SCAN}이 3초가 되는 구간에서 스캐터넷 형성 시간이 큰 결과를 얻었는데, 이는 3.1.2.1의 P_{CON}의 식에 의해 T_{SCAN}이 매우 작으면 P_{CON}의 값이 작아지는 영향에 의한 것으로 추정된다. 반면 3.1.2.1에서 언급했듯이 T_{SEEK}, T_{SCAN} 값이 커지면 하나의 라운드의 지속 시간이 길어져, 한번의 라운드에서 연결이 실패하면 시간 지연이 길어지게 된다. 그림 13의 결과에 의하면 T_{SEEK}이 3.84초, T_{SCAN}이 5초인 경우를 최적화된 값으로 설정한다.

5.1.2 SEEK/SCAN 알고리즘과 제한적 SEEK/SCAN 알고리즘의 성능 비교

3.1.2.2에서 제안한 제한적 SEEK/SCAN 알고리즘과 기존의 SEEK/SCAN 알고리즘의 성능을 비교하기 위한 실험을 수행하였다. 5.1.1의 결과에 준하여 T_{SEEK}을 3.84초, T_{SCAN}을 5초로 설정하고, 노드의 개수를 2개에서 20개까지 늘려가면서 스캐터넷 형성 시간을 측정하여 그림 14와 같은 결과를 얻었다. 제한적 SEEK/SCAN 알고리즘이 기존의 SEEK/SCAN 알고리즘보다 평균적으로 10초 이상의 형성 시간 단축 효과를 가져왔고, 노드의 개수가 많아질수록 형성 시간 차이의 폭이 커짐을 알 수 있다.

본 논문에서는 스캐터넷 알고리즘의 성능을 평가하기 위한 또 다른 기준으로 노드의 개수 N에 따른 스캐터넷 형성 만료 시간인 T_{SC}를 아래와 같이 정의하였다.

$$T_{SC} = 10N[\text{sec}]$$

스캐터넷 형성 실험 중에는 그 형성 시간이 매우 길어지는 경우가 발생하는데, 그 시간이 T_{SC}를 초과할 경우 스캐터넷 형성이 실패한 것으로 간주한다. 이를 바탕

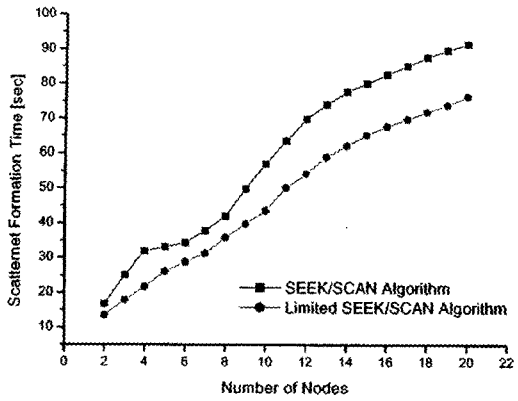

그림 14 두 알고리즘의 스캐터넷 형성 확률 비교

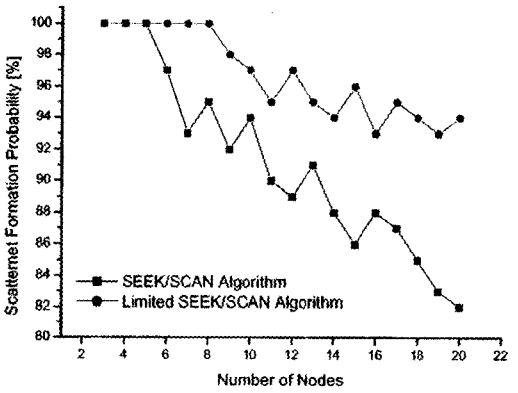

그림 15 두 알고리즘의 스캐터넷 형성 시간 비교

으로 스캐터넷 형성 확률을 측정하여 그림 15와 같은 결과를 얻었다. 그림 15의 결과에 따르면 두 알고리즘 모두 노드의 디바이스의 개수가 많아질수록 형성 확률은 감소하게 된다. 그러나 기존의 SEEK/SCAN 알고리즘에 비해 제한적 SEEK/SCAN 알고리즘이 스캐터넷 형성 확률에 있어서 현저하게 높은 결과를 얻었다.

결과적으로 본 연구에서 제안한 제한적 SEEK/SCAN 알고리즘을 NRS에 적용하였을 경우 형성 시간을 단축시키고, 형성 확률을 높이는 결과를 얻었다.

5.2 스캐터넷 재형성 실험

3.2에서 제안한 노드 추가 및 복구 알고리즘에 대한 성능 실험을 하였다. 추가 알고리즘의 경우 스캐터넷을 구성하는 노드의 개수를 변화시키면서 RING HEAD 노드의 BD_ADDR과 비교하여 그 값이 큰 노드와 작은 노드를 각각 추가하며 선형 구조를 형성할 때까지의 시간을 측정하여 그림 16과 같은 결과를 얻었다. 그림 16에 의하면 노드 추가에 의한 스캐터넷 재형성 시간은 스캐터넷의 노드 수와 무관하게 일정한 결과를 얻었다. 여기서 TAIL로 추가될 때 재형성 시간이 약간 길게 측정된 이유는 3.2.1에서 설명했듯이 HEAD, TAIL로 추가될 때 각각 재형성 알고리즘이 약간 차이가 나는 원인으로 추정된다.

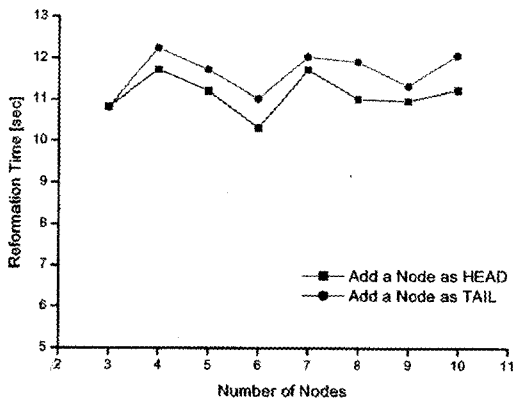

그림 16 노드 추가로 인한 재형성 실험 결과

이번에는 3.2.2에서 제시된 두 가지 복구 알고리즘을 바탕으로 레벨 2,3,4에 대한 복구 시간 및 확률을 실험적으로 측정하였다. 실험은 8개의 노드로 구성된 링형 스캐터넷이 완성된 상태에서 특정 노드의 전원을 제거한 후 스캐터넷이 복구되는 시간과 확률을 측정하였다. 두 알고리즘 모두 N_{RETRY}는 2회로 설정하였다.

표 2,3,4는 각각 레벨2,3,4에 대한 실험 결과이다. 표 2의 결과를 보면 레벨 2에 대해 예약 복귀 노드(RRN) 알고리즘이 DIAC 알고리즘에 비해 복구 시간이 단축되었음을 알 수 있다. 이는 3.2.2에 언급하였듯이 RRN 알고리즘에는 Inquiry 과정이 없기 때문이다. 또 표 2의 RRN 알고리즘에 대한 시간 지연은 대부분 재연결을 위한 Page 과정을 N_{RETRY}번 반복하는 데에서 발생한다. N_{RETRY}값을 줄이면 복구 지연 시간은 더 단축될 수 있다.

표 3의 결과에 의하면 레벨 3에 대해서는 DIAC 알고리즘은 고립 노드의 발생 혹은 복구 노드의 잘못된 연결로 인해 복구 확률이 RRN 알고리즘에 비해 현저히 떨어진다. 반면 RRN 알고리즘은 레벨 2와 비슷한 결과를 레벨 3의 경우에서도 낼 수 있었다. 표 4에서는 레벨 4에 대해서는 DIAC 알고리즘이 레벨 2와 비슷한 결과를 낼 수 있음에 반해 RRN 알고리즘의 경우 순방향으로 2홉 떨어진 노드의 정보만을 저장하고 있으므로 레벨 4의 실험에는 적용할 수 없었다.

결국 레벨 2, 3의 경우에는 RRN 알고리즘을 사용하고, 레벨 4의 경우와 같이 RRN 알고리즘으로 복구가 불가능한 경우가 발생하면 DIAC 알고리즘을 사용하는 방법으로 스캐터넷 복구 알고리즘을 운용할 수 있다.

표 2 레벨 2 복구 실험 결과

알고리즘	복구 지연 시간	복구 확률
DIAC	11.33sec	98%
RRN	8.82 sec	100%

표 3 레벨 3 복구 실험 결과

알고리즘	복구 지연 시간	복구 확률
DIAC	27.21 sec	78%
RRN	10.04 sec	100%

표 4 레벨 4 복구 실험 결과

알고리즘	복구 지연 시간	복구 확률
DIAC	12.29 sec	95%
RRN	-	-

6. 결 론

본 논문에서는 하드웨어 구현을 고려한 노드 링 스캐터넷(Node Ring Scatternet) 알고리즘을 제안하였다. 특히 스캐터넷 초기 형성에 관련되어 기존의 SEEK/SCAN 알고리즘을 개량한 제한적 SEEK/SCAN (Limited SEEK/SCAN) 알고리즘을 제안하였고, 실제 하드웨어 구현 시 시간 지연을 줄이기 위한 T_{SEEK}과 T_{SCAN}에 대한 조건을 명시하고, 이를 실험적으로 검증하였다. 또 노드의 추가 혹은 제거되는 상황에서 스캐터넷 재형성 알고리즘과 관련 노드 추가 알고리즘, DIAC 알고리즘, 예약 노드 복구 알고리즘을 제안하였다.

이상에서 제안된 알고리즘을 하드웨어 구현 시 운용하기 위해 SFMP(Scatternet Formation & Management Protocol)를 상위 프로토콜 스택 상에 설계하였다. 이를 바탕으로 실제 하드웨어 실험을 수행해본 결과 20개의 블루투스 디바이스가 스캐터넷을 형성하고 안정적으로 동작함을 확인하였다. 이는 본 논문에서 실험한 결과이며, 노드의 개수를 추가적으로 늘리는 것에 문제가 없을 것으로 예상된다.

또 제안된 알고리즘을 실제 하드웨어 실험을 통해 그 성능을 평가해본 결과 기존의 유사 알고리즘에 비해 형성 시간이 단축되고, 형성 확률이 증가하는 결과를 얻었다. 특히 노드 추가 혹은 제거에 관련된 재형성 알고리즘이 실제 하드웨어 실험을 통해 안정적인 동작이 되는 결과를 얻었다. 이를 통해 블투투스 스캐터넷이 동적 ad-hoc 네트워크 환경에 지원이 될 수 있는 가능성을 제시할 수 있었다.

본 논문에서 제시된 NRS 알고리즘은 하드웨어 구현이 용이하며, 확장성이 매우 크다는 장점이 있다. 본 논문에서는 스캐터넷 형성에 관련된 부분만 다루었으며, 추후 NRS에서 간략하면서도 효율적인 라우팅 방법을 제안하고자 한다. 또 이번 논문에서 제안된 NRS 알고리즘과 실제 하드웨어 구현을 바탕으로, 향후 무선 센서 네트워크와 같은 실제 응용 분야에 직접 적용하여 하나의 응용 사례를 제시할 예정에 있다.

참 고 문 헌

[1] Bluetooth SIG, "Specification of the Bluetooth System Ver1.1," In *http://www.bluetooth.com*, 2001.

[2] Bluetooth SIG, "Specification of the Bluetooth System Ver1.2," In *http://www.bluetooth.com*, 2003.

[3] P. Bhagwat and A. Segall, "A Routing Vector Method(RVM) for Routing in Bluetooth Scatternet," In *Proceedings of the IEEE International Workshop on Mobile Multimedia Communications (MoMuC)*, pp. 375-379, 1999.

[4] G. Tan, A. Miu, J. Guttag and H. Balakrishnan, "Forming Scatternets from Bluetooth Personal Area Networks," In *MIT Technical Report MIT_LCS-TR-826*, 2001.

[5] C. Lay and K.Y. Siu, "A Bluetooth Scatternet Formation Algorithm," In Proceedings of *the Global Telecommunications Conference(GLOBECOM '01)*, Vol. 5, pp. 2864-2869, 2001.

[6] C. Law, A.K. Mehta and K.Y. Siu, "Performance of a New Bluetooth Scatternet Formation Protocol," In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing*, pp. 182-192, 2001.

[7] G. Zaruba and S. Basagni, "Bluetress-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks," In *Proceedings of the IEEE International Conference on Communicatins (ICC 2001)*, Vol. 1, pp. 273-277, 2001.

[8] C. Petrioli, S. Basagni and I. Chlamtac, "Configuring BlueStars: Multihop Scatternet Formation for Bluetooth Networks," In *Proceedings of the IEEE Transactions on Computers*, Vol. 52, Issue 6, pp. 779-790, 2003.

[9] T.Y. Lin, Y.C. Tseng, K.M. Chang and C.L Tu, "Formation, Routing, and Maintenance Protocols for the BlueRing Scatternet of Bluetooths," In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS '03)*, pp. 313-322, 2003.

[10] H. Zhang, J.C. Hou and L. Sha, "A Bluetooth Loop Scatternet Formation Algorithm," In *Proceedings of the IEEE International Conference on Communications (ICC '03)*, Vol. 2, pp. 1174-1180, 2003.

[11] C.C. Foo and K.C. Chua, "Bluering-Bluetooth Scatternets with Ring Structures," In *Proceedings of the IASTED International Conference on Wireless and Optical Communication (WOC 2002)*, 2002.

[12] S. Basagni, R. Bruno and C. Petrioli, "A Performance Comparison of Scatternet Formation Protocols for Networks of Bluetooth Devices," In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pp. 341-350, 2003.

[13] S. Baatz, M. Frank, C. Kuhl, P. Martini and C. Scholz, "Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme," In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, Vol. 2, pp. 782-790, 2002.

[14] CSR, "Scatternet Support in HCIStack 1.1v16x Firmware," In *CSR Application Note*, 2002.

이 한 욱

1995년 3월~1999년 2월 서울대학교 기계항공공학부 학사. 1999년 3월~2001년 2월 서울대학교 기계항공공학부 석사. 2001년 3월~현재 서울대학교 기계항공공학부 박사과정 재학중. 관심분야는 무선 계측 시스템, 회전체 계측 시스템, 블루투스 스캐터넷 프로토콜

고 상 근

1973년 3월~1978년 2월 서울대학교 기계공학 학사. 1968년 3월~1980년 2월 서울대학교 기계공학 석사. 1984년 3월~1987년 8월 서울대학교 기계공학 박사. 1988년 2월~1989년 2월 스탠포드 대학교 방문 선임 연구원. 1997년 12월~1999년 1월 UCLA 방문 교수. 1989년 3월~현재 서울대학교 기계항공공학부 교수. 관심분야는 메카트로닉스 기반 시스템, 회전체 계측 시스템, 무선 계측 시스템, 카트로닉스(Cartronics)