소규모 분산망을 위한 RNG 기반 스캐터넷 구성 알고리즘

RNG-based Scatternet Formation Algorithm for Small-Scale Ad-Hoc Network

조 정 호*
Chung-Ho Cho

요 약

본 논문에서는 RNG 기반의 스캐터넷 토폴로지 구성, 자기 치유 및 라우팅 경로의 최적화를 위한 알고리즘(RNG-FHR)을 제시하고, ns-2와 블루투스 확장인 blueware 시뮬레이터를 이용하여 알고리즘의 성능을 분석하였다. 소규모 분산망 환경에서 스캐터넷 구성, 노드의 링크 설정 및 자기 치유 관점에서 분석한 결과, 초기에는 이웃노드 발견을 위한 메시지 교환량이 증가하여 지연이 점차 증가하지만 시간이 지남에 따라 이웃노드 발견과 충분한 로컬 정보 교환에 따라 지연이 다소 줄어듦을 알 수 있다. 결국 블루투스 망을 RNG 기반으로 확장하여 소규모의 분산망에 적용하는 경우 소량의 메시지 교환과로컬 정보 교환으로 인해망 구성, 자기 치유 및 라우팅 경로의 최적화가 가능하여 성능과 구현측면에서 충분히 실현 가능함을 알 수 있다.

Abstract

This paper addresses a RNG based scatternet topology formation, self-healing, and routing path optimization for small-scale distributed environment, which is called RNG-FHR(Scatternet Formation, self-Healing and self-Routing path optimization) algorithm. We evaluated the algorithm using ns-2 and extensible Bluetoothsimulator called blueware to show that RNG-FHR does not have superior performance, but is simpler and more practical than any other distributed algorithms from the point of deploying the network in the small-scale distributed dynamic environment due to the exchange of fewer messages and local control. As a result, we realized that even though RNG-FHR is unlikely to be possible for deploying in large-scale environment, it surely can be deployed for performance and practical implementation in small-scale environment.

☞ Keyword : Bluetooth, Scatternet, Blueware, RNG

1. 서 론

블루투스는 단거리이면서 적은 전력을 소모하는 통신에 적합한 표준으로, TDD 마스터-슬레이브 메커니즘을 기본으로 하여 수동적인 망 구성이나 유선망 하부구조 없이 에드 혹 망을 구축할 수 있는 기능을 갖는다. 피코넷은 하나의 마스터 노드와 최대 7개의 슬레이브 노드로 구성되는 그룹으로서 마스터 노드는 다른 슬레이브 노드간 전송을 제어하는 망을 의미한다. 피코넷의 기본 통신은 슬레이브 노드는 홀수 슬롯을 사용하도록 하고 그전 짝수 슬롯은 마스터 노드가 사용하도록 하여 전송 슬롯을 번갈아 줌으로써 이루어진다. 주파수 홉핑은 간섭으로 인한 성능의 저하를 초래하지 않는 가운데 각기 다른 피코넷간 다중 노드의 동시 통신을 위해사용하며, 고밀도의 통신 노드간에 상호 동작 및 다른 피코넷과 독립적으로 통신할 수 있도록 함으로써 다중 피코넷간 연동이 가능하도록 함으로써 피코넷이 오버래핑되는 새로운 개념의 에드 혹 망인 스캐터넷을 탄생시키게 되었다. 자기 치유 및 복구 기능은 노드의 이동성 혹은 노드/링크의 고장으로 인해 새로운 노드의 추가 혹은 기존 노드의 제거시 망을 재

* 정 회 원 : 광주대학교 정보통신학과 부교수
chcho@gwangju.ac.kr
[2007/01/08 투고 - 2007/01/25 심사 - 2007/03/30 심사완료]

구성 가능함을 의미한다. 자체 라우팅 경로의 최적화란 망 전체의 용량과 같은 성능 지표의 요구사항을 만족시키기 위해 라우팅 경로를 최적화시키는 것을 말한다. 본 논문에서는 소규모 분산망 환경에서 RNG(Relative Neighborhood Graph) 기반의 스캐터넷 토폴로지 구성, 자기 치유 및 복구, 홉 기반의 자체 라우팅 최적화를 위한 알고리즘(일명 RNG-FHR: RNG-scatternet Formation, self-Healing and self-Routing path optimization)을 제시하고 성능을 분석한 다음, 망 구축이 현실적으로 가능한지를 분석하였다.

2. 관련 연구

2.1 스캐터넷 구성 알고리즘의 분류

블루투스 피코넷내의 두통신 장치(노드)들이 통신을 할 경우 그중 오직 하나만이마스터 노드의 역할을 수행하고 다른 통신 장치들은 슬레이브 노드로서 역할을 수행한다. 하나의 노드는기껏해야 하나의 피코넷에서 마스터로 동작하며 다른 피코넷에 대해서는 슬레이브로의 역할을 갖는다. 결국 마스터 노드와 슬레이브 노드를 어떻게 결정할 것인지, 새로운 슬레이브 노드는 어느 피코넷에 포함시킬 것인지, 피코넷간을 연결하는 브리지 노드가 몇 개까지의 피코넷을 서비스하도록 할 것인지, 가능한 한 피코넷을 최소한의 수로 구성한다든지, 피코넷 내부 노드간 혹은 피코넷간 통신량을 어떻게 최소화할 것인지, 마스터 노드와 슬레이브 노드의 스케줄링을 얼마나 효율적으로 할 것인지 등은 스케터넷 구성시 성능에 영향을 주는 주요 특성이다.

스케터넷 구성 알고리즘의 분류 기준은 몇가지로 나누어볼 수 있다. 첫째 분류 기준은 연결성이며, 스캐터넷이 연결성(connectivity)을 가진다 함은 스케터넷 구성 초기에 이미 노드간에 연결이 이루어진 상태로서 각 노드는 통신 범위내에 있는 모든 이웃 노드들을알고 있음을 의미한다. 두번째 분류 기준은 노드의 제한적 연결도이며, 이는 각각의 생성된 피코넷의 연결성 제한, 즉 마스터 노드당 최대 7개의 슬레이브 노드와의 연결을 갖는 것을 의미한다. 세번째 분류기준은 단일 홉과 멀티 홉으로 분류된다. 단일 홉 스캐터넷 형성은 실험실내 가전기기나 회의실의 랩탑 컴퓨터와 같이 노드들이 라디오파의 가시거리내에서 동작하는 망을 구성함을 의미한다. 멀티 홉 스캐터넷 형성은 몇몇의 노드들이 라디오파의 가시거리내에서 직접 통신할 수 없고 단지 중간 노드를 경유하여 메시지를 전달하는 망을 구성함을 의미한다[7][8]. 이는 가정이나 사무실 환경과 같은 에드 혹 망에서 군집(clustering) 구조를 이용함으로써 스캐터넷간 연결된 망 구조를 의미한다. 멀티 홉 스캐터넷은 토폴로지 형성에 영향을 주는 정확한 위치 정보를 요구하며 각자의 스캐터넷은 짧은 가시거리내 기기들로 구성된다[9][10]. 네번째 기준은 각 노드가 자신의 기능을발휘하기 위해 어느 이웃 노드까지 알아야 하는가이다

2.2 스캐터넷 구성 알고리즘 설계시 고려 사항

스캐터넷 구성 알고리즘을 설계할 때 고려할 사항은 어떻게 이웃노드를 발견하느냐이다. 즉 두 노드가 어떻게 서로를 발견하고 통신로 설정을 하느냐이다. 대부분의 스캐터넷 구성 알고리즘은 노드 발견시 각 노드가 질의(Inquiry) 모드와 질의 탐지(Inquiry scan) 모드를 랜덤하게 선택하며 모드의 지속 시간(타임 아웃 시간) 길이 역시 랜덤하게 선택된다. 단 모드의 타임아웃 시간은 적절한 시간이내에 단일 홉(hop) 정보를 충분히전달 가능하도록 결정되어야 한다[11]. 노드 발견이 완료되면 마스터노드와 슬레이브 노드간 링크가 설정되며 두 노드는 동일한주파수를 갖게된다. 블루투스 기반의 노드 발견은 비효율적임을 발견한 연구결과도 있다[12]. 연구 결과를 보면 노드의 전체 연결은 상당히 빨리 설정되나 모든 이웃 노드를 알아야 하는 시간이 많이 걸림을 알 수 있다.

두번째 고려 사항은 알고리즘 구성을 중앙집중적으로 하느냐 아니면 분산형태로 하느냐이다. 중앙집중형은 블루투스 망이 인터넷과 같은 고정망에 연결되는 경우 연결점이 중앙집중식 알고리즘을 수행함으로써 모든 노드의 마스터와 슬레이브 역할을 부여하는 방식이다. 그러나 이보다 더 좋은 방법은 중앙노드의 제어가 없이 분산적으로 동작하는 알고리즘을 설계하는 것이다. 분산형 알고리즘은 다시 전역적(globalized) 프로토콜과 지역적(localized) 프로토콜로 나눌 수 있다. 지역적 프로토콜에서는 각 노드가 단지 이웃 노드로 부터의 정보(최대 2 홉 이웃으로 부터의 정보)만을 토대로 구성을 결정한다.

세번째 고려 사항은 노드의자기 치유 기능을 갖는 것이다. 노드가 이동하거나 아니면 새로운 노드가 출현한다거나 아니면 사라질 경우 기존 스케터망은 망구성을 다시 해야 하는데 이를 어떻게 효율적으로 수행하느냐이다. 만일 노드 변경이 단일 홉 혹은 두개의 홉을 갖는 이웃 노드와의 관계만으로 피코넷에 영향을 주는경우 이를 지역적 자기 치유(localized self-healing)라 부르고, 그렇지 않고 2홉 이상의 연속적인 이웃노드와의 정보를 주고받아 연결고리 효과를 내어 자기 치유를 하는 경우 준 지역적(quazi-local) 프로토콜이라 부른다[13].

블루투스 스캐터넷의 링크 구성 과정은 질의 과정(Inquiry Process)과 페이지(Page Process) 두 단계로 구성된다. 질의 과정의 목적은 마스터 노드가 이웃한노드가 존재하는지를 발견하는 과정으로, 이 과정에서 이웃 노드 자신들이 갖는 클럭에 상대적인 정보를 수집한다. 페이지 과정은 질의 과정에서 얻은정보를 이용하여 양방향 주파수 홉핑 통신 채널을 설정한다[3].

본 논문에서는 멀티홉 스캐터넷에 관한 기 연구된 방법 중 소규모망에서 구축 가능한 RNG 기반의 에드 혹 망 구성을 위한 알고리즘을 제안한다. 제안한 알고리즘은 새로운 노드가 추가되거나 노드/링크의 고장으로 인한 노드/링크의 제거로 인한 망의 재구성 기능을 가지며, 최소의 비용을 갖는 최적화된 라우팅 경로를 설정하는 기능을 갖는다.

3. RNG 조건

RNG 알고리즘의 기본 개념은 링크가 발견될 때 이를 그래프에 추가하는 것이다. |AB|를 노드 A와 B간 유클리드 거리로 나타낼 때, RNG에서 두 노드간 링크를 추가하는 필요충분조건은 다른 하나의 노드 C에 대해 |AB| <= max(|BC|, |AC|)를 만족하는 경우이다. 여기서 C는 A와 B의 전송영역에 존재한다(그림 1). RNG 그래프에 링크 AB가 추가된 후, 위 조건을 위배하는 노드 C가 발견되는 경우 RNG는 링크 AB를 그 그래프에서제거한다.

RNG 그래프는 다음과 같은 가정과 특성을 갖는다.

- 각 노드는 토폴로지 그래프내에 있는 이웃한 노드를 알고 있다고 가정한다
- 하나의 노드는 전력 측정과 이웃한 노드와의 일련의 정보교환에 의해 이웃한 노드간 유클리드 거리의 순서를 알고 있다.
- 링크의 추가와 제거는 다른 링크의 추가 혹은 제거에 영향을 주지 않으며, 단지 로컬 정보에만 의존한다.
- 망 전반에 걸쳐 정보를 방송 할 필요가 없으므로 적은 수량의 메시지를 교환하고 MST (Minimum Spanning Tree)와 같은 여느 다른 분산 알고리즘에 비해 구현이 간편하고 용이하다 [6].

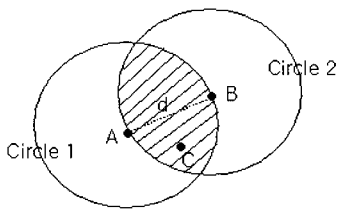


그림 1. 빗금친 부분에 어느 다른 노드가 없을 경우 링

크 AB가 추가된다. 링크 AB가 추가된 후 노드 C가 이 영역에 존재하게 되는 경우 그 링크는 제거된다.

4. RNG 기반의 스캐터넷 구성, 자기 치유 및 라우팅 경로 최적화 알고리즘

RNG 기반 스캐터넷을 구성 알고리즘은 초기화 과정, 노드 발견 과정, 링크 설정과정, RNG 조건에 따른 링크 재설정 과정, 라우팅 최적화과정, 노드 발견 과정의 종료 과정인 6단계로 구성된다. 초기화 단계(Phase 1)에서는 각 노드가 갖는 테이블과 상태를 초기화하는 과정이다. 노드 발견 과정(Phase 2)은 각 노드는 이웃한 노드들을 발견하는 과정으로서 이웃한 하나 이상의 노드를 발견하는 경우 링크 설정 과정(Phase 3)에 명세한 바와 같이 블루투스 명세에 따라 그 중 하나의 이웃 노드와 링크 연결을 시도한다. 링크 설정 과정(Phase 3)에서 일단 마스터 노드(A)로부터 슬레이브 노드(C)로의 물리 링크(주파수 홉핑 연결) 설정이 완료되면 몇가지 규칙에 따라 논리 링크를 설정할 수 있는지 결정한다. 만일 논리링크 설정이 거절되면 설정한 물리링크를 해제한다. 물리링크와 논리 링크 설정 과정이 완료되면 다음 단계인 RNG 조건에 따른 링크 재설정 과정(Phase 4)을 수행하는데 이과정은 노드 A와 C가 자신들이 갖고 있는 NT의 정보를 서로 교환한 다음, 2개의 홉(hop)을 갖는 이웃 노드들에 대한 정보를 기초로 새로운링크를 수락할지 결정하는 과정으로, 이때 RNG 규칙에 따른 링크 재설정 과정을 수행한다. 링크 설정과 RNG 조건을 만족하는 링크 재설정 과정이 이루어지면 그 다음은 홉수 기준의 라우팅 최적화과정(Phase 5)을 수행한다. 이 과정에서 노드A와 C에 대해 HTC가 변경될 경우 그 노드는 이웃 노드에게 변경사항을 알려 이웃 노드들로 하여금 HTC를 갱신하도록 한다. 이웃 노드 역시 자신과 이웃한 노드에게 이를 알려 HTC를 갱신하도록 한다. 이러한 방식으로 모든 노드가 HTC를 갱신하도록 함으로써 스캐터넷 전체에 대한 홉수 기준의 라우팅 최적화를 수행할 수 있다. 모든 노드들이 HTC <1000인 경우에는 더 이상 이웃 노드 발견 과정을 수행하지 않고 종료한다. 만일 새로운 노드가 발견되거나 기존 노드의 이동 혹은 제거시 스캐터넷 재구성을 위한 망의 자기 치유는 앞에서기술한 과정(Phase 2 - Phase 6)을 수행함으로써 이루어진다.

Phase 1 : 각 노드에 대해 초기화를 수행한다.

Piconet_no = [my_bd_addr];

Node_type = Free_node;

Neighbor_Table(NT) = empty vector;

Format of NT is <Bd_addr, Sig_strength, Node_type, Roles, Ch_id, Piconet_no, Hops_to_Controller>;

Bd_addr is the identifier of Bluetooth device;

Sig_strength is the signal strength from the neighbor node;

Role is defined as one of As_Master/As_Slave/Not_Connected;

Ch_id is assigned to each link toward neighbor nodes if a node is connected to its neighbors. Otherwise, not assigned(Ch_id is set as Not_assigned);

Hops_to_Controller(HTC) is to specify the distance to the primary controller;

Neighbor_Piconet_Table(NPT) = empty vector;

NPT is maintained by each master, recording the neighbor piconet combined with local piconet;

Format of NPT is <Bridge_id, Remote_piconet_no>;

Bridge_id is the identifier of the device joining more than one piconet;

Remote_piconet_no is the identifier of the neighbor combined with the local piconet;

```
1. If a node is controller,
2.       then Hops_to_Controller = 0;
3.       else Hops_to_Controller = 1000;
```

Phase 2 : 노드 발견 과정

각 노드는 이웃한 노드들을 발견한 후, 블루투스 명세에 따라 그중 하나의 이웃 노드와 연결을 시도한다.

1. Node A receives inquiry response from node C, meanwhile C creates a new entry for A in its NT based on the inquiry;
2. Node A creates a new entry for C in its NT;
3. Node A makes a link connection to C;

Phase 3 : 링크 설정 과정

일단 마스터 노드 A로부터 슬레이브 노드 C로의 물리 링크(주파수 홉핑 연결)를 설정하면, 아래의 규칙에 따라 논리 링크를 설정할 수 있는지 결정한다. 만일 논리링크 설정이 거절되면 설정한 물리링크를 해제한다.

1. Master A sends RNG_REQ message to the slave C, formatted as <Bd_addr, Node_type, Piconet_no, Hops_to_Controller>
2. Upon receiving the message, the slave C makes a decision whether to accept this connection request. If the node C accepts, it sends back RNG_RESP message as a response and updates local configuration, otherwise disconnect the physical link.
3. Upon receiving the responding message, RNG_RESP, the master A updates local information

Phase 4 : RNG 조건에 따른 링크 재설정 과정

노드 A와 C는 자신들이 갖고 있는 NT의 정보를 서로교환한 다음, 2개의 홉(hop)을 갖는 이웃 노드들에 대한 정보를 기초로 새로운 링크를 수락할지 결정한다. 즉 RNG 규칙에 따른 링크 재설정 과정을 수행한다(그림 8 참조).

1. A and C send their NT(defined as Neighbor_MSG) to each other
2. If (Both of A and C have connected E)
   then {
   Select the weakest link within the triangle;
   If(the weakest link is A->E or C->E)
3. then Disconnect the link, update local information and NT
4. else Disconnect A->C, update local information and NT

Phase 5 : 라우팅 최적화 과정

노드 A와 C에 대해 HTC가 변경될 경우그 노드는 이웃 노드에게 변경사항을 알려 이웃 노드들로 하여금 HTC를 갱신하도록 한다. 이웃 노드 역시 자신과 이웃한 노드에게 이를 알려 HTC를 갱신하도록 한다. 이러한 방식으로 모든 노드가 HTC를 갱신하도록 함으로써 홉수에 따른 라우팅 최적화가 가능하다.

1. If(local Hops_to_Controller for A and C is changed)
   then Broadcast Hop_MSG to their connected neighbors, the format of which is <Bd_addr, Node_type, Piconet_no, Hops_to_Controller>;
2. On receiving Hop_MSG for all nodes
   If(local Hops_to_Controller > Hops_to_Controller in the Hop_MSG + 1)
   then {
   Update NT based on the received Hop_MSG;
   Set local Hops_to_Controller=Hops_to_Controller in the Hop_MSG + 1;
   Generate and broadcast Hop_MSG to all the neighbors;
   }

Phase 6 : 이웃 노드 발견 과정(phase 2 - phase 5)을 종료할 조건을 정한다.

1. If all the node has (Hops_to_Controller < 1000);

2. then Stop discovery;
3. else Continue discovery procedure;
4. We assume that there are six nodes(node 0 to node 5)initially in small-scale scatternet;
5. Node0 is controller which is interfaced to a bus and controls the communication between the other nodes. The other nodes(node1 to node 5) are master or slave node;

5. RNG 기반 스캐터넷 구성 동작 과정

단계 1 : 초기 상태 - 초기화

노드 0는 다른 스캐터넷을 발견하고 이들을 연결하는 기능을 담당하는 제어기이며, 노드 1~노드 5는 어떻게 이웃노드를 발견하느냐에 따라 마스터/슬레이브/마스터-슬레이브 브리지/슬레이브-슬레이브 브리지중의 한 역할을 차지한다.

초기에 노드 1 ~ 노드 5는 초기에 node type= Free_node, role=Not_assigned, , Ch_id = Not_assigned, Hops_to_Controller=1000로 설정한다.

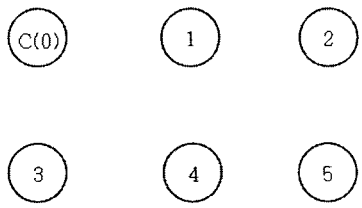


그림 2. 초기 상태

단계 2 : 노드 1의 노드 2 발견

노드 1은 이웃테이블(NT: Neighbor Table)에 노드 2에 관한 엔트리와 신호세기를 만든다. 이때 노드 1이 갖는 NT의 엔트리는 <002, 3, Free_node, Not_Connected, Not_assigned, [1], 1000> 이다. 노드 1은 노드 2에 <001, Free_node, [1], 1000>내용의 연결요청 메시지(RNG_REQ) 보내면, 노드 2는 현재 Free_node 상태이므로 이를 수락하고 노드 2의 NT에 노드 1에 관한 엔트리를 <001, 3, Master_Node, As_Master, Not_assigned, [1], 1000>내용으로 갱신한 다음, <002, Slave_Node, 1, 1000>내용의 응답 메시지(RNG_RESP)를 보낸다. RNG_RESP 메시지를 수신한 노드 1은 노드 2에 대해 <002, 3, Slave_Node, 0, [1], 1000>내용으로 NT엔트리를 갱신하고, local Node_type = Master_Node and local Piconet_no = 1로 자신의 로컬 정보를 갱신한다. 노드 1과 노드 2는 NT에 있는 엔트리 정보를 Neighbor_MSG로 교환한다. 이때 RNG조건이 만족되지 않으므로 이 메시지를 받은 후 아무런 조치를 취하지 않는다. 또한 노드 1과 노드 2는 Hops_to_Controller=1000으로서 변화가 없으므로 Hop_MSG를 방송하지 않는다. 모든 노드가 Hops_to_Controller < 1000를 만족하지 않으므로 노드 발견 절차는 지속된다.

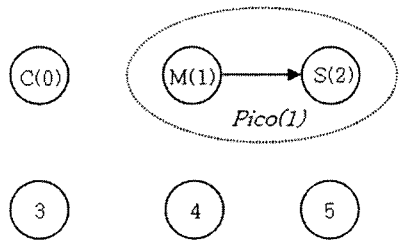


그림 3. 노드 1이 노드 2를 발견한 경우의 토폴로지

단계 3 : 노드 4의 노드 2 발견

노드 4는 자신의 NT에 노드 2에 관한 엔트리를만든다. 시나리오에서 신호 세기는 1로 한다. 노드 4에 있는 엔트리 내용은 is <002, 1.5, Free_node, Not_Connected, Not_assigned, [4], 1000> 이다. 노드 4는 노드 2에 <004, Free_node, [4], 1000>내용의 연결요청 메시지 RNG_REQ를 보낸다. 노드 2는 자신의 NT에 <004, 1.5, Free_node, Not_Connected, Not_assigned, [4], 1000>내용으로 노드 4에 대한 엔트리를 갱신하고, local Piconet_no(=1)가 4와 같지 않으므로 요청을 수락

한 다음, 노드 4에 대한 엔트리를 <004, 1.5, Master_Node, As_Master, 1, [4], 1000>로 갱신하고, 노드 2를 슬레이브-슬레이브 브리지(S-S_Bridge)로 등록하고, 새로운 연결(4->2)을 노드 1에 보고한 후, 노드 4에 <002, S-S_Bridge, [1,4], 1000>의 내용의 응답 메시지(RNG_RESP)를 보낸다. 노드 1은 이웃 피코넷 테이블(NPT:Neighbor Piconet Table)에 엔트리 <002, 4>을 추가하여 갱신한다. 노드 2로부터 RNG_RESP를 수신하면 노드 4는 NT를 <002, 1.5, S-S_Bridge, As_Slave, 0, [1,4], 1000>내용으로 갱신하고 Node_type = Master_Node and local Piconet_no = 4와 같은 로컬 정보를 갱신한다. 노드 4는 수신된 RNG_RESP 메시지가 S-S_Bridge로부터 받았으므로 NPT를 <002, 1>로 갱신한다. 노드 2와 노드 4는 서로 Neighbor_MSG를 교환한 다음, 노드 2와 노드 4가 동일한 피코넷상의 이웃이 아니므로 RNG 조건이 성립하지는 않으므로 두 노드는 Neighbor_MSG를 서로 수신한 후에도 아무 행동을 취하지 않고 Hops_to_Controller 역시 변경되지 않았으므로 Hop_MSG를 방송하지 않는다. 모든 노드가 Hops_to_Controller < 1000를 만족하지 않으므로 노드 발견 절차는 계속된다.

SS : Slave-Slave bridge

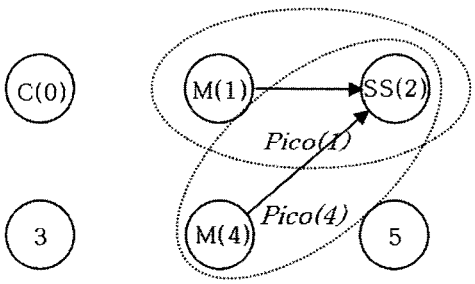


그림 4. 노드 4가 노드 2를 발견한 경우의 토폴로지

단계 4 : 노드 0의 노드 3 발견

노드 0는 자신의 NT에 노드 3에 대한 엔트리를만든다. 이 경우 노드 3으로 부터의 신호 세기가 2라 가정한다. 이 경우 노드 0의 NT에 있는 엔트리는 <000, 2, Free_node, Not_Connected, Not_assigned, [0], 1000>이다. 노드 0는 노드 3에 <000, Free_node, [0], 1000>내용으로 RNG_REQ를 보낸다. 노드 3은 자신의 NT에 노드 0에 대한 엔트리를 <000, 2, Master_Node, As_Master, Not_assigned, [0], 0>로 갱신하고 로컬 정보를 Hops_to_Controller=1, accept=YES로 설정하고노드 0에 <003, Slave_Node, [0], 1>내용으로 RNG_RESP 메시지를 보내어 연결을 수락한다. 노드 3으로부터 RNG_RESP 메시지를 수신한 노드 0는 자신의 NT에 있는 엔트리를 <003, 2, Slave_Node, As_Slave, Not_assigned, [0], 1>로 갱신한 다음 Node_type = Master_Node and local Piconet_no = 0의 로컬 정보를 갱신한다.

노드 0와 노드 3는 서로 Neighbor_MSG를 교환한 다음, 두 노드가 동일한 피코넷상의 이웃이 아니므로 RNG 조건이 성립하지는 않으므로 두 노드는 Neighbor_MSG를 서로 수신한 후에도 아무 행동을 취하지 않고 Hops_to_Controller 역시 변경되지 않았으므로 Hop_MSG를 방송하지 않는다. 모든 노드가 Hops_to_Controller < 1000를 만족하지 않으므로 노드 발견은 지속된다.

CM : Controller with Master node type
HTC of node 1 to node 5 except node 3 is 1000

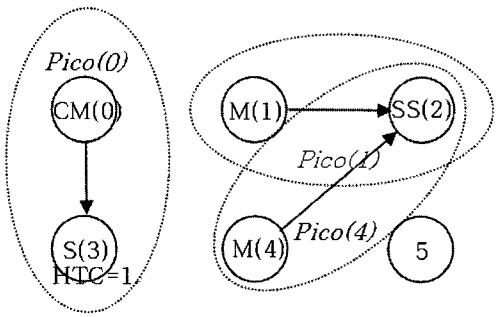


그림 5. 노드 0가 노드 3를 발견한 경우의 토폴로지

단계 5 : 노드 0의 노드 4 발견

노드 0는 자신의 NT에 노드 4에 대한 새로운 엔트리를 만든다. 이 경우 노드 4로부터의 신호 세기는 1.5라 가정한다. 노드 0의 NT에 있는 엔트리는 <004, 1.5, Free_node, Not_Connected, Not_assigned, [0], 1000>가 된다. 노드 0는 노드 4에 <004, Master_Node, [0], 1000>내용의 연결 요청 메시지(RNG_REQ)를 보낸다. 노드 4는 자신의 NT에 노드 0에 대한 엔트리를 <000, 1.5, Master_Node, Not_Connected, Not_assigned, [0], 1000>로 갱신하고, 노드 4가 Master_Node이므로 연결 요청을 수락한다. node_type=MS_Bridge and pico_no=[0,4]로 로컬 정보를 갱신하고, 노드 4를 M-S_Bridge로 등록한 다음, 노드 0에 <004, M-S_Bridge, [0,4], 1000> 내용의 연결 요청에 대한 응답 메시지(RNG_RESP)를보낸다. 노드 0는 로컬 Hops_to_Controller=0로 갱신하고, NPT에 엔트리 <004, 4>를 추가하여 갱신한다. 노드 4로부터 RNG_RESP를 수신한 노드 0는 NT를 <004, 1.5, M-S_Bridge, As_Slave, 1, [0,4], 1>로 갱신하고 로컬 정보를 Node_type = Master_Node and local Piconet_no = 0로 갱신한다. 노드 0는 M-S bridge로부터 RNG_RESP 메시지를 수신하기 때문에 NPT를 <004, 4>로 갱신한다. 노드 0와 노드 4는 Neighbor_MSG를 서로 교환하고, 두 노드가 동일한 피코넷에 존재하지 않으므로 RNG 조건이 성립하지 않는다. 따라서 두 노드는 Neighbor_MSG를 수신한 후에 아무 행동을 취하지 않는다. 노드 4의 Hops_to_Controller가 1000에서 1로 변경되었으므로 노드 4는 Hop_MSG를 노드 0와 노드 2에 <004, MS_Bridge, [0,4],1>내용으로 방송한다. 노드 4로부터 Hop_MSG를 수신한 노드 2는 로컬 정보를 HPC(Hops_to_Controller)=2로 갱신한 다음 <002, SS_Bridge, [0,4], 2>내용의 Hop_MSG를 이웃한 노드에 방송한다. 노드 2로부터Hop_MSG를 수신한 노드 1은 로컬 정보를 HPC=3로 갱신하고, 노드 1은 Hop_MSG를 이웃 노드에 방송한다. 노드 0가 노드 4로부터Hop_MSG를 받고, 노드 4가 노드 2로부터Hop_MSG를 받으며, 노드 2가 노드 1으로부터 Hop_MSG를 받으면 HTC <= (HTC in the Hop_MSG + 1) 이므로 그들은 아무 행위도 취하지 않는다. 즉 각자의 로컬 정보인 HTC를 갱신하지 않는다. 모든 노드가 Hops_to_Controller < 1000를 만족하지 않으므로 노드 발견 절차는 지속된다.

MS : Master-Slave bridge
HTC of node 3 and node 4 is 1
HTC of node 1, node 2 and node 5 is 1000

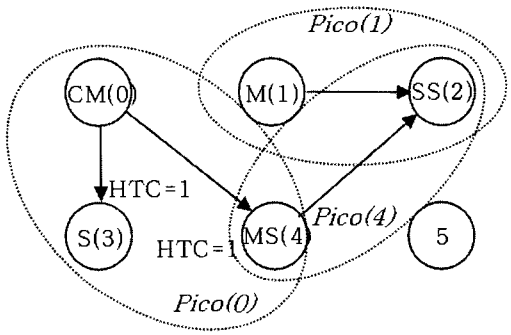


그림 6. 노드 0가 노드 4를 발견한 경우의 토폴로지

단계 6 : 노드 3이 노드 4로부터 Inq_Response를 수신(노드 3이 노드 4 발견)

노드 3은 자신의 NT에 노드 4에 대해 <004, 3, Free_node, Not_Connected, Not_assigned, [3/Not_assigned], 1000>내용의 새로운 엔트리를 만든다. 이때 노드 4로 부터의 신호 세기는 3이라 가정한다. 노드 3은 노드 4에 <003, Slave_Node, [0], 1> 내용의 RNG_REQ 메시지를 보내어 연결 요청을 한다. 노드 4는 자신의 NT에 노드 3에 대한 엔트리를 <003, 3, Slave_Node, Not_Connected, 1, [0], 1000>내용으로 갱신한다. 노드 4는 MS_Bridge이고 자신의 로컬 정보인 Piconet_no (=0)이 수신한 RNG_REQ의 Piconet_no(=0)와 일치하므로 새로운 링크를 거절하고 링크(3->4)를 해제한다.

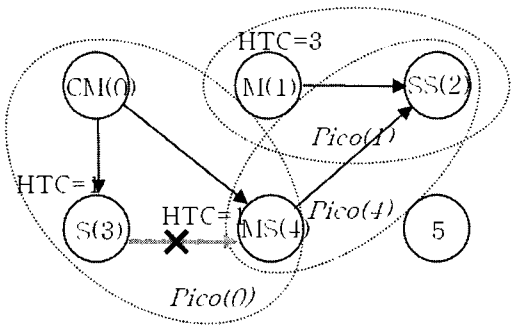


그림 7. 노드 3이 노드 4를 발견한 경우의 토폴로지

단계 7: 노드 4가 노드 1으로부터 Inq_Response 수신(노드 4가 노드 1 발견)

노드 4는 자신의 NT에 노드 1의 <001, 2, Free_node, Not_Connected, Not_assigned, [4]/Not_assigned, 1000>내용의 새로운 엔트리를 만든다. 이 경우 노드 1으로부터의 신호 세기는 2라 가정한다. 노드 4는 노드 1에 <004, MS_Bridge, [0,4], 1>내용의 연결 요청 메시지인 RNG_REQ을 보낸다. 노드 1은 자신의 NT에 노드 4에 대한 엔트리를 <004, 2, MS_Bridge, As_Master, 1, [0,4], 1>내용으로 갱신한다. 노드 1은 마스터 노드가 아니므로 연결 요청을 수락하고 로컬 정보를 accept=YES, type=MS_Bridge, Piconet_no=[1,4]로 갱신한다. 그림 7에서 노드 1의 로컬 정보가 HTC=3이며 수신한 RNG_REQ의 HTC=1이므로 노드 1은 조건 (Hops_to_Controller of node 4 + 1 < local Hops_to_Controller of node 1)이 1+1 <= 3이 되어 로컬 정보인 HTC를 갱신한다. 노드 1은 NPT에 새로운 엔트리 <001, 4>를 추가하여 갱신하고, 노드 4에 <001, MS_Bridge, [1,4], 2>내용의 RNG_RESP를 보낸다. 노드 4는 <001, 2,MS_Bridge, As_Slave, 2, [2,4], 2>내용의 RNG_RESP를 토대로 NT를 갱신하고, 로컬 정보를 node_type=MS_Bridge, Piconet_no=[0,4]로 갱신한 다음. 노드 4는 RNG_RESP를 MS_Bridge인 노드 1으로부터 받았으므로 <001, 1>내용의 새로운 엔트리를 추가하여 NPT를 갱신한다. 노드 4는 MS_Bridge로 변했으므로 노드 4를 브리지로 등록하기 위해 노드 0에 새로운 연결을 알린다. 노드 1과 노드 4는 서로 Neighbor_MSG를 교환한다. 이들 노드는 노드 2를 동일한 이웃 노드로 가지므로 가장 약한 신호 세기를 갖는 링크(4->2)를 선택한다. 노드 4는 링크(4->2)를 연결할 책임을 지며, NT를 노드 4와 노드 2는 서로 NT에 Not_Connected로 기록한 다음, 로컬 정보를 Piconet_no=[1], HTC=1000, 노드 2 type=slave로 갱신한다. 노드 1은 HTC가 변경되었으므로 Hop_MSG를 노드 4와 노드 2에 방송한다. 노드 2는 수신한 Hop_MSG를 토대로 <001, 3, MS_Bridge, As_Master, 0, [1], 2>내용으로 NT를 갱신하고, 로컬정보인 HTC=1000이며 수신한 Hop_MSG메시지의 HTC=2이므로 HTC=3으로 갱신한다. 노드 4는 노드 1으로부터 Hop_MSG를 수신하지만 NT가 이미 갱신된 상태이므로 아무 행위를 취하지 않는다. 노드 2가 노드 1에 Hop_MSG를 방송하면 노드 1은 이를 수신하나아무것도 변경되지 않는다. 노드 5의 HPC=1000이므로 발견 절차는 지속된다. 라우팅 경로의 최적화와 RNG 조건은 그림 8의 삼각형 음영지역에 적용된다. 노드 4에서 노드 2까지의 링크는 이 영역에서 가장 약한 신호 세기를 나타내므로 이 링크는 해제된다.

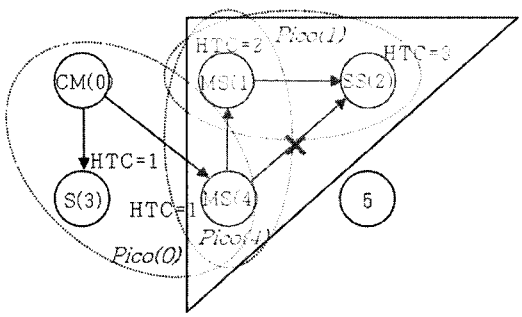


그림 8. 노드 4가 노드 1을 발견한 경우의 토폴로지

단계 8 : 노드 2가 노드 5로부터 Inq Response 수신 (노드 2의 노드 5 발견)

노드 2는 자신의 NT에 노드 5에 대해 <001, 2, Free_node, Not_Connected, Not_assigned, Not_assigned, 1000> 내용의 새로운 엔트리를 만든다. 노드 2는 노드 5에 <002, Slave, [1], 3>내용의 RNG_REQ 메시지를 보내어 연결 요청한다. 노드 5는 <002, 2, Slave, Not_assigned, [1], 3>내용의 새로운 엔트리를 NT에 추가한다. 노드 5는 type=free이므로 연결 요청을 받아들이고 로컬 정보를 node_type=Slave, Piconet_no=2로 한 다음, <002, 2, MS_Bridge, Master, 0, [1,2],3> 내용의 새로운 엔트리를 NT에 추가하여 갱신한다. 노드 5의 로컬정보인 HTC=1000와 수신한 RNG_REQ 메시지의 HTC=3이므로 로컬 HTC를 4로 갱신한다. 노드 5는 <005, Slave, [2], 4>내용의 RNG_RESP 메시지를 노드 2에 보낸다. 이 메시지를 받은노드 2는<005, 2, Slave, As_Slave, 1, [2],4>내용을 추가하여 NT를 갱신하고 로컬 정보를 node_type= MS_Bridge, Piconet_no=[1,2]로 갱신한다. 노드 2는 MS_Bridge가 되므로 노드 1에 노드 2를 브리지로 등록하도록 보고한다. 노드 1은 <002,2>를 추가하여 NPT를 갱신한다. 노드 2와 노드 5는 Neighbor_MSG를 서로 교환하나 RNG 조건을 만족하지 않으므로 아무 행위도 취하지않는다. 노드 5는 로컬 정보인 HTC가 변경되므로 Hop_MSG를 노드 2에 방송한다. 이를 수신한 노드 2는 이미 NT가 갱신된 상태이므로 아무 행위를 취하지 않는다. 모든 노드가 Hop_to_Controller < 1000 조건을 만족하므로 더 이상 노드 발견 절차를 수행하지 않고 종료한다.

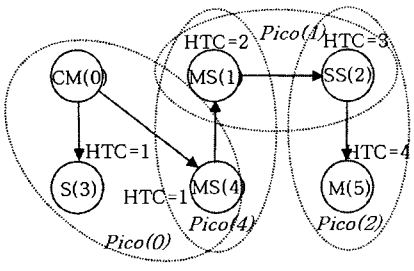


그림 9. 노드 2가 노드 5를 발견한 경우의 토폴로지

6. 성능 분석

제안한 알고리즘에 대한 성능평가를 위해 Godfrey Tan[1][5]에 의해 개발된 Blueware를 이용하였다. Blueware는 다양한 스캐터넷 구성과 링크 스케줄링을 위한 블루투스의 확장이다[4]. 참고문헌 [3]에서 제시한 TSF과 비교하여 본 논문에서 제시한 RNG-FHR알고리즘의 성능을 분석하였다. 성능분석은 스캐터넷 구성, 링크 설정과 안정된 스캐터넷에 도달하기까지의 지연시간을 중심으로 분석하였다. 모든 실험에서 노드는 랜덤한 클럭이 할당되고, 모든 실험 결과 데이터는 30번의 시행의 평균치이다. TSF와 RNG-FHR알고리즘에 대한 시뮬레이션 환경이 완전히 동일하지는 않는 관계로 둘간의 정확한 성능비교에는 데는 약각의 오차가 있을 수 있다.

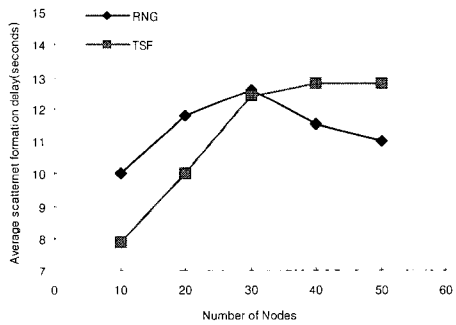


그림 10. Average scatternet formation delay

그림 10에서 본 바와 같이 RNG-FHR알고리즘에서는 스캐터넷 구성에 소요되는 지연이 노드 수가 30이 될때까지는 증가하지만 그 이상이 될 경우 지연이 점차 줄어듦을 알 수 있다. 이는 초기 토폴로지를 구성하는데 이웃 노드와의 정보 교환이 빈번히 발생하기 때문이다. 일단 토폴로지가 구성된 후 이동 혹은 고장으로 인해 새로운 노드가 추가되거나 제거될 시 이웃 노드에 관한 정보만를 토대로 하는 로컬 정보에만 의존함으로써 다른 노드나 링크에 영향을 주지 않는다. 즉,

비교적 적은 수량의 메시지를 사용하고 스캐터넷 재구성이 로컬 정보 교환에 의해 간결하게이루어 진다고 볼 수 있다.

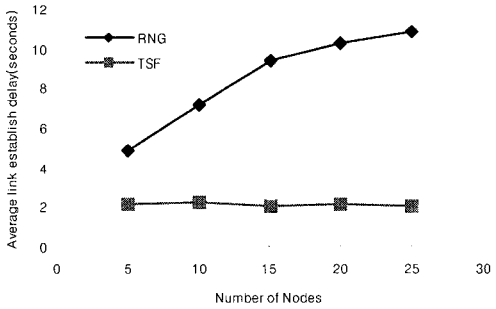


그림 11. Average link establish delay

그림 11은 평균 링크 설정 지연은 기존 토폴로지에 새로운 노드를 추가하는데 걸리는 시간을 보여주고 있다. 노드 수가 증가함에 따라 노드간 링크 설정에 소요되는 지연이 로그 함수적으로 증가하지만 그증가율은 노드수가 증가함에 따라 점차 둔화됨을 알 수 있다. 그 이유는 초기에 각 노드간 메시지 교환량과 이웃 노드를 발견하기 위한 충분한 정보를 확보하기 때문이라고 본다.

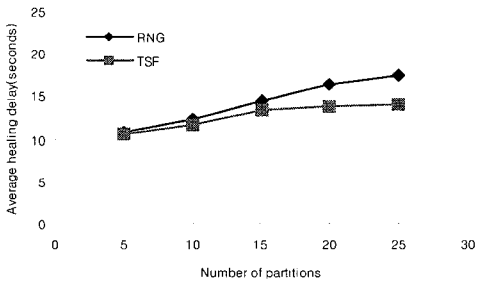


그림 12. Average self-healing delay

그림 12는 새로운 노드가 제거될 경우 토폴로지의 자유 치유에 소요되는 지연 시간을 구한 것이다. 만일 어느 노드가 토폴로지에서 제거될 때 스캐터넷은 더 작은 피코넷으로 분할될 수 있다. 이 경우 제어기 노드는 스캐터넷을 치유하는 동안 각각의 피코넷을 연결시키려 할 것이다. 치유에 따른 지연은 망의 분할 수 증가에 따라 로그 함수적으로 증가함을 알 수 있다. 초기에 비해 토폴로지의 분할 수가 많아짐에 따라 지연 증가율이 다소 둔화됨을 알 수 있는데 이는 적은 수량의 교환 메시지와 이웃한 노드에 대한 로컬 정보를 유지하고 있기 때문이다. TSF의 경우, 노드 발견과 분할된 피코넷의 합병이 제어기와 루트 노드에 제한되고 모든 노드가 전파 거리내에 존재해야 하기 때문에 모든 분할을 치유하지 못할 수도 있다.

7. 결론

블루투스는 피코넷에서 스캐터넷으로 확장되어 발전하고 있다. 피코넷은 하나의 마스터 노드와 최대 7개의 슬레이브 노드로 구성되는 그룹으로서 마스터 노드는 다른 슬레이브 노드간 전송을 제어하는 망을 일컫는다. 주파수 홉핑으로 인해 고밀도의 통신 노드간에 상호 동작 및 다른 피코넷과독립적으로 통신할 수 있도록 함으로써 다중 피코넷간 연동이 가능하게 되었다. 그러나 멀티홉 스캐터넷에 관한 연구 중 소규모망에서 구축 가능한 RNG 기반의 에드 혹 망 구성을 위한 알고리즘에 관한 연구가 미흡하여, 토폴로지 구성, 새로운 노드가 추가되거나 노드/링크의 고장으로 인한 노드/링크의 제거로 인한 망의 재구성, 최소의 비용을 갖는 최적화된 라우팅 경로를 설정하는 기능을 수행하는 알고리즘에 관한 연구가 필요하다.

본 논문에서는 RNG 조건을 만족하는 스캐터넷 구성, 노드의 이탈 및 제거시 토폴로지의 자기 치유 및 라우팅 경로의 최적화를 위한 알고리즘인 RNG-FHR을 제시하였다. RNG-FHR은 로컬 정보만을 의존하고 로컬 정보를 그래프 전반에걸쳐 방송하지 않으며, 소량의 메시지 교환만으로 노드

발견, 토폴로지 구성, 자기 치유 및 라우팅 경로 최적화를 실현할 수 있어 구현이 용이한 잇점을 지닌다.

소량의 메시지 교환과 로컬 정보 의존만으로 소규모 망에 적용할 경우 실현 가능한 성능을 보임을 알 수 있다. 스캐터넷 구성과 자기 치유측면에서 TSF와 거의 비슷한 지연 시간을 보임을 알 수 있다. 그러나 링크 설정측면에서는 RNG-FHR이 TSF에 비해 다소 차이를 보임을 알 수 있다.

따라서 블루투스 스캐터넷은 노드 수가 너무 많아지면 성능의 확연한 저하와복잡도 증가로 인해 대규모 망에 경우 현실적으로 적용하는데 어려움이 크나 소규모 분산망에 적용할 경우 실현 가능한 대안이 될 것으로 보며 향후 성능 개선과 QoS를 고려할 경우의 스케터넷 구성에 관한 연구가 필요하다고 본다.

참 고 문 헌

[1] Blueware: Bluetooth simulator for ns. http://nms.lcs.mit.edu/projects/blueware/, http://nms.csail.mit.edu/projects/blueware/software

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

[3] G. Tan, A. Miu, H. Balakrishnan, and J. Guttag. An Efficient Scatternet Formation Algorithm for Dynamic Environments. In IASTED International Conference on Communications and Computer Network (CCN02), Cambridge, MA, Nov. 2002.

[4] G. Tan and J. Guttag. A Locally Coordinated Scatternet Scheduling Algorithm. In The 27th Annual IEEE Conference on Local Computer Networks(LCN), Tampa, FL, Nov. 2002.

[5] G.Tan. Blueware: Bluetooth Simulator for *ns*. MIT Technical Report, MIT-LCS-TR-866, Cambridge, MA, October, 2002.

[6] Evangelos Vergetis, Roch Guerin, Saswati Sarkar, and Jacob Rank. Can Bluetooth Succeed as a Large-Scale Ad Hoc Networking Technology?. In IEEE Journal on Selected Areas in Communications, Vol. 23, No. 3, March 2005.

[7] Xiang-Yang Li, Yu Wang, Peng-Jun Wan, Wen-Zhan Song, and Ophir Frieder. Localized Low-Weight Graph and Its Applications in Wireless Ad Hoc Networks. In IEEE INFOCOM 2004.

[8] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, and Richard LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. In IEEE INFOCOM 2001.

[9] Chiara Petrioli, Stefano Basagni, and Imrich Chlamtac. Configuring BlueStars : Multihop Scatternet Formation for Bluetooth Networks. In IEEE Transactions on Computers, Vol. 52, No. 6, June 2003.

[10] Xiang-Yang Li, Ivan Stojmenivic, and Yu Wang. Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Scatternet Formation. In IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 4, April 2004.

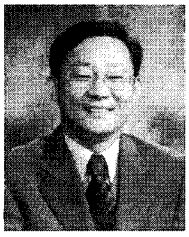
[11] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, Distributed topology construction of Bluetooth personal area networks, Proc. IEEE INFOCOM, 2001.

[12] S. Basagni, R. Bruno and C. Petrioli, A performance comparison of scatternet formation protocols for networks of Bluetooth devices, *Proc. IEEE PerCom*, Texas, Mar. 2003, 341-350.

[13] X.-Y. Li, I. Stojmenovic, Y. Wang, Partial Delaunay triangulation and degree limited

localized Bluetooth scatternet formation, Proc. AD-HOC NetwOrks and Wireless (ADHOC-NOW), Toronto, September 20-21, 2002, 17-32; IEEE Trans. Parallel and Distributed Systems, 15, 4, April 2004.

◐ 저 자 소 개 ◑



조 정 호(Chung-Ho Cho)
1984년 전남대학교 계산통계학과 졸업(학사)
1986년 전남대학교 대학원 전산학과 졸업(석사)
1996년 전남대학교 대학원 전산학과 졸업(박사)
1988~1997 한국전자통신연구소 이동통신기술연구부 선임연구원
2005~2006 미국 University of Arizona, Visiting Professor
1997~현재 광주대학교 정보통신학과 부교수
관심분야 : 무선이동인터넷, 모바일 QoS, 무선 PAN
E-mail : chcho@gwangju.ac.kr